

दिल्ली विश्वविद्यालय
UNIVERSITY OF DELHI

Bachelor of Science (Hons.) Computer Science

(Effective from Academic Year 2019-20)



Revised Syllabus as

approved by

Academic Council

Date:

No:

Executive Council

Date:

No:

**Applicable for students registered with Regular Colleges, Non Collegiate
Women's Education Board and School of Open Learning**

List of Contents

Preamble	2
1. Introduction to Programme B.Sc. (H) Computer Science	3
2. Learning Outcome-based Curriculum Framework in Programme B.Sc. (H) Computer Science	3
2.1 Aims of Bachelor Degree Programme in B.Sc. (H) Computer Science	3
3. Graduate Attributes in B.Sc. (H) Computer Science	3
4. Qualification Descriptors for Graduates B.Sc. (H) Computer Science	4
5. Programme Learning Outcomes for in B.Sc. (H) Computer Science	4
6. Structure of in B.Sc. (H) Computer Science.	5
6.1 Credit Distribution for B.Sc. (H) Computer Science	5
6.2 Semester-wise Distribution of Courses.	6
7. Courses for Programme B.Sc. (H) Computer Science	10
8. Acknowledgement	127

Preamble

The objective of any programme at Higher Education Institute is to prepare their students for the society at large. The University of Delhi envisions all its programmes in the best interest of their students and in this endeavour it offers a new vision to all its Under-Graduate courses. It imbibes a Learning Outcome-based Curriculum Framework (LOCF) for all its Under Graduate programmes.

The LOCF approach is envisioned to provide a focused, outcome-based syllabus at the undergraduate level with an agenda to structure the teaching-learning experiences in a more student-centric manner. The LOCF approach has been adopted to strengthen students' experiences as they engage themselves in the programme of their choice. The Under-Graduate Programmes will prepare the students for both, academia and employability.

Each programme vividly elaborates its nature and promises the outcomes that are to be accomplished by studying the courses. The programmes also state the attributes that it offers to inculcate at the graduation level. The graduate attributes encompass values related to well-being, emotional stability, critical thinking, social justice and also skills for employability. In short, each programme prepares students for sustainability and life-long learning.

The new curriculum of B.Sc. (H) Computer Science offers to develop theoretical foundations in computer science to build computational thinking, analytical, and problem solving skills. The programme builds a base for entry level jobs in information technology and prepares the students for higher studies in the area of Computer Science/Applications. The course aims to produce skilled graduates with a creative mind-set who can recognize a computational problem either in IT industry or society, and develop effective solutions. Understanding the needs of society and societal obligations are instilled in courses related to AI and Information security.

The students develop expertise in programming skills using contemporary programming languages used by software industry. It covers core computer science topics like computer systems architecture, data structures, computer networks, operating systems, computer graphics, algorithms, software engineering, database management, theory of computation, artificial intelligence, and information security. The mode of learning shall be a blend of the formal and the inquiry based methods, with special focus on practical and projects.

The University of Delhi hopes the LOCF approach of the programme B.Sc. (H) Computer Science will help students in making an informed decision regarding the goals that they wish to pursue in further education and life, at large.

1. Introduction to Programme B.Sc. (H) Computer Science

The B.Sc. (H) Computer Science programme is designed to develop analytical & computational thinking, and problem solving skills. It covers the core computer science topics like computer systems architecture, data structures, computer networks, operating systems, computer graphics, algorithms, software engineering, database management, theory of computation, artificial intelligence, and information security. The programme builds a base for entry level jobs in information technology and prepares the students for higher studies in the area of Computer Science/Applications.

2. Learning Outcome-based Curriculum Framework for B.Sc. (H) Computer Science programme

2.1 Aims of Bachelor Degree Programme in B.Sc. (H) Computer Science

- i. Develop theoretical foundations in computer science.
 - ii. Develop expertise in programming skills using high level programming languages.
 - iii. Develop skills to design, implement and document the solutions for computational problems.
 - iv. Develop soft skills to work effectively in a team to solve a problem.
 - v. Develop the ability to use state of the art technologies.
 - vi. Inculcating the understanding of the needs of society and the importance of societal obligations.
-

3. Graduate Attributes in B.Sc. (H) Computer Science

Disciplinary knowledge

Ability to build (either independently or by joining higher academic program) on of the core computer science concepts learnt in the course.

Ability to apply the core computer science concepts to solve the problems in the IT industry.

Problem solving

Graduates are equipped with skills to solve the computational problems at their workplace and for the society.

Cooperation/Team work

Graduates demonstrate competence to use communication skills to participate or lead a team for a new initiative or for solving an existing problem.

Communication Skills

Graduates demonstrate effective communication and presentation skills while interacting with professional peers and in the society.

Scientific reasoning

Given a problem, the graduates will be able to analyse it, suggest solutions, and critically evaluate the solutions proposed by others.

Professional Ethics: Graduates follow ethical principles and commitment to professional ethics, accountability and responsibilities.

4. Qualification Descriptors for Graduates B.Sc. (H) Computer Science

- i. Demonstrate coherent knowledge and understanding of the logical organization of a digital computer, its components and working. Understanding of the time and space complexities of algorithms designed to solve computational problems.
- ii. Demonstrate programming skills in high level language and an ability to learn a new programming language without substantial effort.
- iii. Apply knowledge of logical skills to identify and analyse problems and issues, and seek solutions to real-life problems. For example, creating mobile applications, database applications, and educative computer games.
- iv. Enhanced communication and leadership abilities and ability to work and learn in team environment.
- v. Understand the needs of society and sensitivity to societal obligations

5. Programme Learning Outcomes for B.Sc. (H) Computer Science

- i. Ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.

- ii. Ability to design, implement, and evaluate a computer-based system, process, component, or program to solve the given problem.
- iii. Ability to communicate effectively through oral and written means.
- iv. Ability to work in a team to achieve a common goal

6. Structure of B.Sc. (H) Computer Science

6.1 Credit Distribution for B.Sc. (H) Computer Science

Course	*Credits	
	Theory + Practical	Theory+ Tutorial
<u>I. Core Course</u>		
(14 Papers)	14 X 4 = 56	14 X 5 = 70
Core Course Practical/ Tutorial*		
(14 Papers)	14 X 2 = 28	14 X 1 = 14
<u>II. Elective Course</u>		
(8 Papers)		
A.1. Discipline Specific Elective	4 X 4 = 16	4 X 5 = 20
(4 Papers)		
A.2. Discipline Specific Elective		
Practical/ Tutorial*	4 X 2 = 8	4 X 1 = 4
(4 Papers)		
B.1. Generic		
Elective/Interdisciplinary	4 X 4 = 16	4 X 5 = 20
(4 Papers)		
B.2. Generic Elective Practical/		
Tutorial*	4 X 2 = 8	4 X 1 = 4
(4 Papers)		

*** Optional Dissertation or project work in place of one Discipline Specific Elective paper (6 credits) in 6th Semester**

III. Ability Enhancement Courses

1. Ability Enhancement Compulsory

(2 papers of 4 credit each)

$4 \times 2 = 8$

$4 \times 2 = 4$

Environmental Science

English/MIL Communication

2. Skill Enhancement Elective

$4 \times 2 = 8$

$4 \times 2 = 8$

(2 papers of 4 credit each)

Total Credit

140

140

* wherever there is a practical there will be no tutorial and vice-versa

6.2 Semester-wise Distribution of Courses.

SE ME ST ER	DISCIPLINE SPECIFIC CORE COURSE (DSC) (14)	Ability Enhancement Compulsory Course (AECC) (2)	Skill Enhanceme nt Course (SEC) (2)	Elective: Discipline Specific (DSE) (4)	Elective: Generic (GE) (6)
I	Programming Fundamentals using C++				GE-1
	Computer System Architecture				
II	Programming in JAVA				GE-2
	Discrete Structure				
III	Data Structures		SEC-1		GE-3
	Operating System				
	Computer Networks				
IV	Design and Analysis of Algorithms		SEC-2		GE-4
	Software Engineering				
	Database Management Systems				
V	Internet Technologies			DSE-1	
	Theory of Computation			DSE-2	
VI	Artificial Intelligence			DSE-3	
	Computer Graphics			DSE-4	

Semester	COURSE OPTED	COURSE NAME	CREDITS
I			
	Core Course-I	Programming Fundamentals using C++	4
	Core Course-I Practical/Tutorial	Programming Fundamentals using C++ Lab	2
	Core Course-II	Computer System Architecture	4
	Core Course –II Practical/Tutorial	Computer System Architecture Lab	2
	Generic Elective-I	GE – 1	4/5
	Generic Elective-I Practical/Tutorial		2/1
II			
	Core Course-III	Programming in Java	4
	Core Course –III Practical/Tutorial	Programming in Java Lab	2
	Core Course-IV	Discrete Structure	4
	Core Course –IV Practical/Tutorial	Discrete Structure Tutorial	2
	Generic Elective- 2	GE – 2	4/5
	Generic Elective- 2 Practical/Tutorial		2/1
III			
	Core Course – V	Data Structures	4
	Core Course –V Practical/Tutorial	Data Structures Lab	2
	Core Course - VI	Operating System	4
	Core Course –VI Practical/Tutorial	Operating System Lab	2
	Core Course - VII	Computer Networks	4
	Core Course –VII Practical/Tutorial	Computer Networks Lab	2
	Skill Enhancement Course-1	SEC – 1	4
	Generic Elective – 3	GE- 3	4/5
	Generic Elective - 3 Practical/Tutorial		2/1
IV			
	Core Course – VIII	Design and Analysis of Algorithms	4
	Core Course –VIII Practical/Tutorial	Design and Analysis of Algorithms Lab	2
	Core Course-IX	Software Engineering	4
Core Course –IX Practical/Tutorial	Software Engineering Lab	2	

	Core Course-X	Database Management Systems	4
	Core Course –X Practical/Tutorial	Database Management Systems Lab	2
	Skill Enhancement Course-2	SEC – 2	4
	Generic Elective – 4		4/5
	Generic Electives - 4 Practical/Tutorial		2/1
V	Core Course-XI	Internet Technologies	4
	Core Course –XI Practical/Tutorial	Internet Technologies Lab	2
	Core Course-XII	Theory of Computation	5
	Core Course –XII Practical/Tutorial	Theory of Computation Tutorial	1
	Discipline Specific Elective-1	DSE-1	4
	Discipline Specific Elective-1 Practical/Tutorial	DSE-1 Lab	2
	Discipline Specific Elective-2	DSE-2	4
	Discipline Specific Elective-2 Practical/Tutorial	DSE- 2 Lab	2
VI			
	Core Course-XIII	Artificial Intelligence	4
	Core Course –XIII Practical/Tutorial	Artificial Intelligence Lab	2
	Core Course-XIV	Computer Graphics	4
	Core Course –XIV Practical/Tutorial	Computer Graphics Lab	2
	Discipline Specific Elective-3	DSE-3	4
	Discipline Specific Elective-3 Practical/Tutorial	DSE-3 Lab	2
	Discipline Specific Elective-4	DSE-4	4
Discipline Specific Elective-4 Practical/Tutorial	DSE-4 Lab	2	
Total Credits			140

Discipline Specific Core Papers (DSC): (Credit: 06 each)

(1 period/ week for tutorials or 4 periods/week of practical)

1. BHCS01 Programming Fundamentals using C++
2. BHCS02 Computer System Architecture
3. BHCS03 Programming in JAVA
4. BHCS04 Discrete Structure
5. BHCS05 Data Structures
6. BHCS06 Operating System
7. BHCS07 Computer Networks
8. BHCS08 Design and Analysis of Algorithms
9. BHCS09 Software Engineering

- 10. BHCS10 Database Management Systems
- 11. BHCS11 Internet Technologies
- 12. BHCS12 Theory of Computation
- 13. BHCS13 Artificial Intelligence
- 14. BHCS14 Computer Graphics

Discipline Specific Elective Papers: (Credit: 06 each)
(DSE-1, DSE-2, DSE-3, DSE-4)

DSE-1 (Choose any one)

- a) BHCS15A Data Analysis and Visualization
- b) BHCS15B System Programming
- c) BHCS15C Combinatorial Optimization

DSE – 2 (Choose any one)

- a) BHCS16A Digital Image Processing
- b) BHCS16B Microprocessors

DSE – 3 (Choose any one)

- a) BHCS17A Information Security
- b) BHCS17B Data Mining
- c) BHCS17C Advanced Algorithms

DSE – 4 (Choose any one)

- a) BHCS18A Machine Learning
- b) BHCS18B Deep Learning
- c) BHCS18C Unix Network Programming
- d) BHCS18D Project Work/ Dissertation

Other Discipline (Four papers of any one discipline) – GE 1 to GE 4

Skill Enhancement Courses (Credit: 04 each)
(SEC – 1, SEC – 2)

SEC -1(Choose any one)

- a) BHCS19A Web Design and Development
- b) BHCS19B Programming in Python

SEC – 2(Choose any one)

- a) BHCS20A Android Programming
- b) BHCS20B Introduction to R Programming

Note:

1. There will be one batch of 10-15 students for practical classes. The size of tutorial group for papers without practical is recommended to be 8-10 students.
 2. Each practical will carry 50 marks including 25 marks for continuous evaluation and 5 marks for the oral viva.
 3. Colleges are advised and encouraged to conduct the practical using Free and Open Source Software (FOSS)
 4. At least two questions have to be compulsorily attempted in the final practical examination.
 5. Softcopy of all the practical must be maintained by each student for each practical paper.
 6. Discipline specific core and elective courses (DSC and DSE) are to be taught as 4 Hrs theory and 4 Hrs practical per week. In case the course has tutorials, it is to be taught as 5 Hrs theory and 1 Hr. tutorial per week
 7. Skill enhancement courses (SEC) are to be taught as 2 Hrs theory and 4 Hrs practical per week.
 8. Practical given for the courses are only indicative, and by no means exhaustive. Instructor may add more complex problems in laboratory depending on the ability of the students.
-

7. Courses for Programme B.Sc. (H) Computer Science

Programming Fundamentals using C++ (BHCS01) Discipline Specific Core

Course - (DSC)

Credit: 06

Course Objective

This course is designed to develop structured as well as object-oriented programming skills using C++ programming language. The course not only focuses on basic C++ constructs but also covers object-oriented programming features in-depth, namely Encapsulation, Abstraction, Inheritance and Polymorphism for writing efficient codes.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Explain significance of object oriented paradigm

2. Solve programming problems using object oriented features.
3. Handle external files as well as exceptions.
4. Reuse classes to create new classes.
5. Handle exceptions in programs.

Detailed Syllabus

Unit 1

Introduction to C++: Overview of Procedural Programming and Object-Oriented Programming, Using main () function, Header Files, Compiling and Executing Simple Programs in C++.

Unit 2

Programming Fundamentals: Data types, Variables, Operators, Expressions, Arrays, Keywords, Naming Convention, Decision making constructs (if, switch), Looping (for, while, do...while), Type Casting, Input-output statements, Functions, Command Line Arguments/Parameters.

Unit 3

Object Oriented Programming: Overview of Abstraction, Encapsulation, Inheritance, and Polymorphism. Creating Classes and objects, Modifiers and Access Control, Constructors, Implementation of Inheritance (Single and multilevel), Implementation of Polymorphism (Function Overloading and Operator Overloading, Function Overriding).

Unit 4

Pointers and References: Static and dynamic memory allocation, Pointer and Reference Variables, Pointers vs. References, Implementing Runtime polymorphism using pointers and references.

Unit 5

Exception and File Handling: Using try, catch, throw, throws and finally; Nested try, creating user defined exceptions, File I/O Basics, File Operations.

Practical

1. Write a program to compute the sum of the first n terms of the following series:

$$S = 1 - 1 / (2 ^ 2) + 1 / (3 ^ 3) - \dots 1 / (n ^ n)$$

where ^ is exponentiation.

The number of terms n is to be taken from user through command line. If command line

- argument is not found then prompt the user to enter the value of n.
2. Write a program to remove the duplicates from an array.
 3. Write a program that prints a table indicating the number of occurrences of each alphabet in the text entered as command line arguments.
 4. Write a menu driven program to perform following operations on strings (without using inbuilt string functions):
 - a) Show address of each character in string
 - b) Concatenate two strings.
 - c) Compare two strings
 - d) Calculate length of the string (use pointers)
 - e) Convert all lowercase characters to uppercase
 - f) Reverse the string
 5. Write a program to merge two ordered arrays to get a single ordered array.
 6. Write a program to search a given element in a set of N numbers using Binary search
 - (i) with recursion (ii) without recursion.
 7. Write a program to calculate GCD of two numbers (i) with recursion (ii) without recursion.
 8. Create Matrix class. Write a menu-driven program to perform following Matrix operations:
 - a) Sum
 - b) Product
 - c) Transpose
 9. Define a class Person having name as a data member. Inherit two classes Student and Employee from Person. Student has additional attributes as course, marks and year and Employee has department and salary. Write display() method in all the three classes to display the corresponding attributes. Provide the necessary methods to show runtime polymorphism.
 10. Create a class Triangle. Include overloaded functions for calculating area. Overload assignment operator and equality operator.
 11. Write a program to read two numbers p and q. If q is 0 then throw an exception else display the result of p/q.
 12. Rewrite Matrix class of Q8 with exception handling. Exceptions should be thrown by the functions if matrices passed to them are incompatible and handled by main() function.

13. Create a class Student containing fields for Roll No., Name, Class, Year and Total Marks. Write a program to store 5 objects of Student class in a file. Retrieve these records from file and display them.
14. Copy the contents of one text file to another file, after removing all whitespaces.

References

1. Forouzan & Gilbert (2012). *Computer Science: A Structured Approach Using C++*. Cengage Learning.
2. Schildt, H. (2003). *C++: The Complete Reference*. 4th edition. Tata McGraw-Hill.

Additional Resources

1. Balaguruswamy, E. (2017). *Object Oriented Programming with C++ (7th ed.)*. McGraw Hill Education.
2. Kanetkar, Y. P. (2015). *Let us C++*. 2nd edition. BPB Publishers.
3. Prata, S. (2015). *C++ Primer Plus* 6th edition. Pearson Education India.
4. Stroustrup, B. (2013). *The C++ Programming Language*. 4th Edition. Pearson Education.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Introduction to C++: Overview of Procedural Programming and Object-Oriented Programming, Using main () function, Header Files, Compiling and Executing Simple Programs in C++
2	Data types, Variables , Operators, Expressions, Arrays, Keywords, Naming Convention, Type Casting, Input-output statements
3	Decision making constructs (if, switch), Looping (for, while, do...while)

4	Functions, Command Line Arguments/Parameters
5 – 9	Overview of Abstraction, Encapsulation, Inheritance, and Polymorphism. Creating Classes and objects, Modifiers and Access Control, Constructors, Inheritance (Single and multilevel), Polymorphism (Function Overloading, Operator Overloading, Function Overriding)
10-12	Static and dynamic memory allocation, Pointer variables, Reference Variables, Pointers vs. References, Runtime polymorphism using pointers and references
13-15	Exception and File Handling: Using try, catch, throw, throws and finally; Nested try, creating user defined exceptions, File I/O Basics, File Operations

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Procedural and Object Oriented programming, abstraction, inheritance, polymorphism, pointers, exception and file handling

Computer System Architecture (BHCS02) Discipline Specific Core Course - (DSC) Credit: 06

Course Objective

This course introduces the students to the fundamental concepts of digital computer organization, design and architecture. It aims to develop a basic understanding of the building blocks of the computer system and highlights how these blocks are organized together to architect a digital computer system.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Design Combinational Circuits using basic building blocks. Simplify these circuits using Boolean algebra and Karnaugh maps. Differentiate between combinational circuits and sequential circuits.
2. Represent data in binary form, convert numeric data between different number systems and perform arithmetic operations in binary.
3. Determine various stages of instruction cycle and describe interrupts and their handling.
4. Explain how CPU communicates with memory and I/O devices.
5. Simulate the design of a basic computer using a software tool

Detailed Syllabus

Unit 1

Digital Logic Circuits: Logic Gates, truth tables, Boolean Algebra, digital circuits, combinational circuits, sequential circuits, circuit simplification using Karnaugh map, Don't Care Conditions, flip-flops, characteristic tables

Unit 2

Digital Components: Half Adder, Full Adder, Decoders, Multiplexers, Registers and Memory Units

Unit 3

Data Representation and Basic Computer Arithmetic: Number system, complements, fixed and floating point representation. Alphanumeric representation. Addition, subtraction.

Unit 4

Basic Computer Organization and Design: Common Bus system, instruction codes, instruction format, instruction set completeness, Sequence Counter, timing and control, instruction cycle, memory reference instructions and their implementation using arithmetic, logical, program control, transfer and input output micro-operations, interrupt cycle.

Unit 5

Central Processing Unit: Micro programmed Control vs Hardwired Control, lower level programming languages, Instruction format, accumulator, general register organization, stack organization, zero-address instructions, one-address instructions, two-address instructions, three-address instructions, Addressing Modes, RISC, CISC architectures, pipelining and parallel processing.

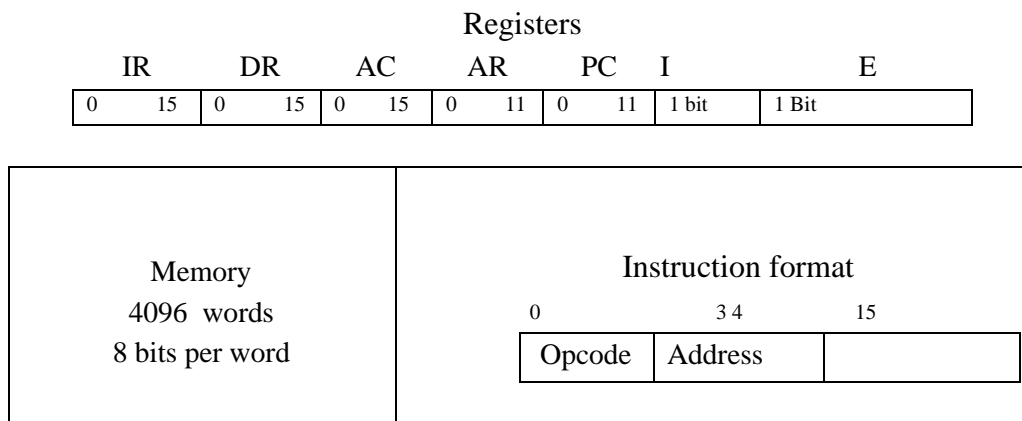
Unit 6

Memory Organization and Input-Output Organization: Input-Output Organization: Peripheral Devices, I/O interface, I/O vs. Memory Bus, Programmed I/O, Interrupt-Driven I/O, Direct Memory Access

Practical

(Use Simulator – CPU Sim 3.6.9 or any higher version for the implementation)

1. Create a machine based on the following architecture :



Basic Computer Instructions

Memory Reference		Register Reference		
Symbol	Hex	Symbol	Hex	
AND	0xxx	CLA	E800	
ADD	2xxx	CLE	E400	
LDA	4xxx	CMA	E200	
STA	6xxx	CME	E100	
BUN	8xxx	CIR	E080	
		CIL	E040	
ISZ	Cxxx	INC	E020	
AND_I	1xxx	SPA	E010	
ADD_I	3xxx	SNA	E008	
LDA_I	5xxx	SZA	E004	
STA_I	7xxx	SZE	E002	
BUN_I	9xxx	HLT	E001	
ISZ_I	Dxxx			

Refer to Chapter-5 of reference 1 for description of instructions.

Design the register set, memory and the instruction set. Use this machine for the assignments of this section.

2. Create a Fetch routine of the instruction cycle.
3. Write an assembly program to simulate ADD operation on two user-entered numbers.
4. Write an assembly program to simulate SUBTRACT operation on two user-entered numbers.
5. Write an assembly program to simulate the following logical operations on two user-entered numbers.

1. AND
2. OR
3. NOT
4. XOR
5. NOR
6. NAND

6. Write an assembly program to simulate MULTIPLY operation on two user-entered numbers.
7. Write an assembly program for simulating following memory-reference instructions.

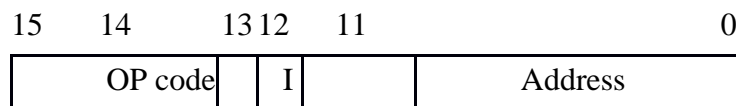
1. ADD
2. LDA
3. STA
4. BUN
5. ISZ

8. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution:

1. CLA
2. CMA
3. CME
4. HLT

9. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution:
1. INC
 2. SPA
 3. SNA
 4. SZE
10. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution:
1. CIR
 2. CIL
11. Write an assembly program that reads in integers and adds them together; until a negative non-zero number is read in. Then it outputs the sum (not including the last number).
12. Write an assembly program that reads in integers and adds them together; until zero is read in. Then it outputs the sum.
13. Create a machine for the following instruction format:

Instruction format



The instruction format contains a 3-bit opcode, a 1-bit addressing mode and a 12-bit address.

Write an assembly program to simulate the machine for addition of two numbers with I= 0 (Direct Address) and address part = 082. The instruction to be stored at address 022 in RAM, initialize the memory word with any decimal value at address 082. Determine the contents of AC, DR, PC, AR and IR in decimal after the execution.

14. Simulate the machine for the memory-reference instruction referred in above question with I= 1 (Indirect Address) and address part = 082. The instruction to be stored at address 026 in RAM. Initialize the memory word at address 082 with the value 298. Initialize the memory word at address 298 with operand 632 and AC with 937. Determine the contents of AC, DR, PC, AR and IR in decimal after the execution.

15. The instruction format contains 3 bits of opcode, 12 bits for address and 1 bit for addressing mode. There are only two addressing modes, I = 0 is direct addressing and I = 1 is indirect addressing. Write an assembly program to check the I bit to determine the addressing mode and then jump accordingly.

References

1. Mano, M. (1992). *Computer System Architecture*. 3rd edition. Pearson Education.

Additional Resources

1. Mano, M. (1995). *Digital Design*. Pearson Education Asia.
2. Null, L., & Lobur, J. (2018). *The Essentials of Computer Organization and Architecture*. 5th edition. (Reprint) Jones and Bartlett Learning.
3. Stallings, W. (2010). *Computer Organization and Architecture Designing for Performance* 8th edition. Prentice Hall of India.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1 – 2	Unit 1 - Introduction: Digital Logic Gates, Flipflops and their characteristic table, Logic circuit simplification using Boolean Algebra and Karnaugh Map, Don't Care conditions. Combinational Circuits, Sequential Circuits.
3 – 4	Unit 2 - Digital Components: Decoders, Encoders, Multiplexers, Binary Adder, Binary Adder-Subtractor, Binary Incrementer, Registers and Memory Units
5 – 6	Unit 3 - Data Representation:

	Binary representation of data, representation of alpha data, representation of numeric data in different number systems, conversion between number systems, complements, representation of decimal numbers, representation of signed and unsigned numbers, addition and subtraction of signed and unsigned numbers and overflow detection.
7 - 11	Unit 4 - Operations and Control: Arithmetic and logical micro-operations, micro programmed control vs. hardwired control, instruction format, instruction set completeness, timing and control, instruction cycle, memory reference instructions and their implementation using arithmetic, logical, program control, transfer and input output micro operations, interrupt cycle.
12 - 13	Unit 5 - Instructions: Instruction format illustration using single accumulator organization, general register organization and stack organization, Addressing Modes, zero-address instructions, one-address instructions, two-address instructions and three-address instructions,
14 - 15	Unit 6 - Peripheral Devices: I/O interface, I/O vs. Memory Bus, Isolated I/O, Memory Mapped I/O, Direct Memory Access

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Combinational and sequential circuits, memory organization, computer organization, CPU design, parallelism.

Programming in JAVA (BHCS03) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

This course adds to the basic programming language skills acquired by the student in earlier semesters. The students are exposed to the advanced features available in Java such as exception handling, file handling, interfaces, packages and GUI programming.

Course Learning Outcomes

On successful completion of the course the student will be

1. Implement Exception Handling and File Handling.
2. Implement multiple inheritance using Interfaces.
3. Logically organize classes and interfaces using packages.
4. Use AWT and Swing to design GUI applications.

Detailed Syllabus

Unit 1

Review of Object Oriented Programming and Java Fundamentals: Structure of Java programs, Classes and Objects, Data types, Type Casting, Looping Constructs.

Unit 2

Interfaces Interface basics; Defining, implementing and extending interfaces; Implementing multiple inheritance using interfaces Packages Basics of packages, Creating and accessing packages, System packages, Creating user defined packages

Unit 3

Exception handling using the main keywords of exception handling: try, catch, throw, throws and finally; Nested try, multiple catch statements, creating user defined exceptions

Unit 4

File Handling Byte Stream, Character Stream, File I/O Basics, File Operations

Unit 5

AWT and Event Handling: The AWT class hierarchy, Events, Event sources, Event classes, Event Listeners, Relationship between Event sources and Listeners, Delegation event model, Creating GUI applications using AWT.

Unit 6

Swing Introduction to Swing, Swing vs. AWT, Hierarchy for Swing components, Creating GUI applications using Swing.

Practical

1. Design a class Complex having a real part (x) and an imaginary part (y). Provide methods to perform the following on complex numbers:
 1. Add two complex numbers.
 2. Multiply two complex numbers.
 3. toString() method to display complex numbers in the form: $x + i y$
2. Create a class TwoDim which contains private members as x and y coordinates in package P1. Define the default constructor, a parameterized constructor and override toString() method to display the co-ordinates. Now reuse this class and in package P2 create another class ThreeDim, adding a new dimension as z as its private member. Define the constructors for the subclass and override toString() method in the subclass also. Write appropriate methods to show dynamic method dispatch. The main() function should be in a package P.
3. Define an abstract class Shape in package P1. Inherit two more classes: Rectangle in package P2 and Circle in package P3. Write a program to ask the user for the type of shape and then using the concept of dynamic method dispatch, display the area of the appropriate subclass. Also write appropriate methods to read the data. The main() function should not be in any package.
4. Create an exception subclass UnderAge, which prints "Under Age" along with the age value when an object of UnderAge class is printed in the catch statement. Write a class exceptionDemo in which the method test() throws UnderAge exception if the variable age passed to it as argument is less than 18. Write main() method also to show working of the program.
5. Write a program to implement stack. Use exception handling to manage underflow and overflow conditions.
6. Write a program that copies content of one file to another. Pass the names of the files through command-line arguments.
7. Write a program to read a file and display only those lines that have the first two characters as '/' (Use try with resources).
8. Write a program to create an Applet. Create a frame as a child of applet. Implement mouseClicked(), mouseEntered() and mouseExited() events for applet. Frame is visible

when mouse enters applet window and hidden when mouse exits from the applet window.

9. Write a program to display a string in frame window with pink color as background.
10. Write a program to create an Applet that has two buttons named “Red” and “Blue”. When a button is pressed the background color of the applet is set to the color named by the button’s label.
11. Create an applet which responds to KEY_TYPED event and updates the status window with message (“Typed character is: X”). Use adapter class for other two events.
12. Create an applet with two buttons labeled ‘A’ and ‘B’. When button ‘A’ is pressed, it displays your personal information (Name, Course, Roll No, College) and when button ‘B’ is pressed, it displays your CGPA in previous semester.
13. Write a program that creates a Banner and then creates a thread to scrolls the message in the banner from left to right across the applet’s window.
14. Rewrite the applet programs using Swing.

References

1. Schildt, H. (2018). *Java: The Complete Reference*. 10th edition. McGraw-Hill Education.

Additional Resources:

1. Balaguruswamy E. (2014). *Programming with JAVA: A Primer*. 5th edition. India: McGraw Hill Education
2. Horstmann, C. S. (2017). *Core Java - Vol. I – Fundamentals* (Vol. 10). Pearson Education
3. Schildt, H., & Skrien, D. (2012). *Java Fundamentals - A Comprehensive Introduction*. India: McGraw Hill Education.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Review of Object Oriented Programming and Java Fundamentals Structure of Java programs, Classes and Objects, Data types, Type Casting, Looping Constructs
2	Interfaces Interface basics; Defining, implementing and extending interfaces; Implementing multiple inheritance using interfaces
3	Packages Basics of packages, Creating and accessing packages, System packages, Creating user defined packages
4	Exception Handling : Using the main keywords of exception handling: try, catch, throw, throws and finally; Nested try, Multiple catch statements, Creating user defined exceptions
5	File Handling: Byte Stream, Character Stream, File I/O Basics, File Operations
6-9	AWT and Event Handling The AWT class hierarchy ,Events, Event sources, Event classes, Event Listeners, Relationship between Event sources and Listeners, Delegation event model, Creating GUI applications using AWT, Creating GUI applications using AWT
10-15	Swing: Introduction to Swing, Swing vs. AWT, Hierarchy for Swing components, Creating GUI applications using Swing, Creating GUI applications using Swing

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Objects and classes, interfaces, exceptional handling, file handling

Discrete Structures (BHCS04) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

The course aims to introduce the students to Boolean algebra, sets, relations, functions, principles of counting, and growth functions so that these concepts may be used effectively in other courses.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Define mathematical structures (relations, functions, sequences, series, and graphs) and use them to model real life situations.
2. Understand (trace) and construct simple mathematical proofs using logical arguments.
3. Solve class room puzzles based on counting principles.
4. Compare functions and relations with respect to their growth for large values of the input.

Detailed Syllabus

Unit 1

Introduction: Sets - finite and infinite sets, uncountable infinite sets; functions, relations, properties of binary relations, closure, partial ordering relations; counting - Pigeonhole Principle, permutation and combination; mathematical induction, Principle of Inclusion and Exclusion.

Unit 2

Growth of Functions: asymptotic notations, summation formulas and properties, bounding summations, approximation by integrals.

Unit 3

Recurrence: recurrence relations, generating functions, linear recurrence relations with constant coefficients and their solution, recursion trees, Master Theorem

Unit 4

Graph Theory: basic terminology, models and types, multi-graphs and weighted graphs, graph representation, graph isomorphism, connectivity, Euler and Hamiltonian Paths and Circuits, planar graphs, graph coloring, Trees, basic terminology and properties of Trees, introduction to spanning trees.

Unit 5

Propositional Logic: logical connectives, well-formed formulas, tautologies, equivalences, Inference Theory

Practical

1. Write a Program to create a SET **A** and determine the cardinality of SET for an input array of elements (repetition allowed) and perform the following operations on the SET:
 - a) ismember (a, A): check whether an element belongs to set or not and return value as true/false.
 - b) powerset(A): list all the elements of power set of A.

2. Create a class SET and take two sets as input from user to perform following SET Operations:
 - a) Subset: Check whether one set is a subset of other or not.
 - b) Union and Intersection of two Sets.
 - c) Complement: Assume Universal Set as per the input elements from the user.
 - d) Set Difference and Symmetric Difference between two SETS
 - e) Cartesian Product of Sets.
3. Create a class RELATION, use Matrix notation to represent a relation. Include functions to check if the relation is Reflexive, Symmetric, Anti-symmetric and Transitive. Write a Program to use this class.
4. Use the functions defined in Ques 3 to check whether the given relation is:
 - a) Equivalent, or
 - b) Partial Order relation, or
 - c) None
5. Write a Program to implement Bubble Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.
6. Write a Program to implement Insertion Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.
7. Write a Program that generates all the permutations of a given set of digits, with or without repetition. (For example, if the given set is {1,2}, the permutations are 12 and 21). (One method is given in Liu)
8. Write a Program to calculate Permutation and Combination for an input value n and r using recursive formula of ${}^n C_r$ and ${}^n P_r$.
9. For any number n, write a program to list all the solutions of the equation $x_1 + x_2 + x_3 + \dots + x_n = C$, where C is a constant ($C \leq 10$) and $x_1, x_2, x_3, \dots, x_n$ are nonnegative integers using brute force strategy.
10. Write a Program to accept the truth values of variables x and y, and print the truth table of the following logical operations:

a) Conjunction	f) Exclusive NOR
b) Disjunction	g) Negation
c) Exclusive OR	h) NAND
d) Conditional	i) NOR
e) Bi-conditional	
11. Write a Program to store a function (polynomial/exponential), and then evaluate the

polynomial. (For example store $f(x) = 4n^3 + 2n + 9$ in an array and for a given value of n , say $n = 5$, evaluate (i.e. compute the value of $f(5)$).

12. Write a Program to represent Graphs using the Adjacency Matrices and check if it is a complete graph.

13. Write a Program to accept a directed graph G and compute the in-degree and out-degree of each vertex.

14. Given a graph G , write a Program to find the number of paths of length n between the source and destination entered by the user.

15. Given an adjacency matrix of a graph, write a program to check whether a given set of vertices $\{v_1, v_2, v_3, \dots, v_k\}$ forms an Euler path / Euler Circuit (for circuit assume $v_k = v_1$).

16. Given a full m -ary tree with i internal vertices, Write a Program to find the number of leaf nodes.

References

1. Mohapatra, & Liu, C. L. (2012). *Elements of Discrete mathematics*. 4th edition. McGraw Hill Education.
2. Rosen, K. H. (2011). *Discrete Mathematics and Its Applications*. 7th edition. Tata McGraw Hill Education.

Additional Resources

1. Albertson, M. O., & Hutchinson, J.P., (1988). *Discrete Mathematics with Algorithms*. John Wiley and Sons.
2. Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (2009). *Introduction to algorithms*. 3rd edition. MIT Press.
3. Hein, J. L. (2015). *Discrete Structures, Logic, and Computability*. 4th edition. Jones and Bartlett Learning.
4. Hunter, D. J. (2011). *Essentials of Discrete Mathematics*. 2nd edition. Jones and Bartlett Learning

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-3	Sets Finite and infinite sets, uncountable infinite sets; functions, relations, properties of binary relations, closure, partial ordering relations, pigeonhole principle, permutation and combination, induction, inclusion exclusion
4-5	Growth of Functions Asymptotic notations, summation formulas and properties, summation formulas and properties (contd.), bounding summations, approx. by integrals
6-8	Recurrences Recurrence relations, generating functions, linear recurrence relations with constant coefficients and their solution, recursion trees, Master's Theorem
9-13	Graph Theory Basic terminology, models and types, multigraphs and weighted graphs, graph representation, graph isomorphism, connectivity, Euler and Hamiltonian Paths and Circuits, planar graphs, graph coloring, Trees, basic terminology and properties of Trees, introduction to spanning trees.
14-15	Propositional Logic Logical connectives, well-formed formulas, tautologies, equivalences, inference theory

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

recurrence, trees and graphs, combinatorics, inductive and deductive reasoning, asymptotic complexity.

Data Structures (BHCS05) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

This course aims at developing the ability to use basic data structures like array, stacks, queues, lists, trees and hash tables to solve problems. C++ is chosen as the language to understand implementation of these data structures.

Course Learning Outcomes

At the end of the course, students will be able to:

1. Implement and empirically analyse linear and non-linear data structures like Arrays, Stacks, Queues, Lists, Trees, Heaps and Hash tables as abstract data structures. (RBT L2/3)
2. Write a program, choosing a data structure, best suited for the application at hand. (RBT L3/4)
3. Re-write a given program that uses one data structure, using a more appropriate/efficient data structure (RBT L4)
4. Write programs using recursion for simple problems. Explain the advantages and disadvantages of recursion.(RBT L2/L3)
5. Identify Ethical Dilemmas.

Detailed Syllabus

Unit 1

Arrays: single and multi-dimensional arrays, analysis of insert, delete and search operations in arrays (both linear search and binary search), implementing sparse matrices, applications of arrays to sorting: selection sort, insertion sort, bubble sort, comparison of sorting techniques via empirical studies. Introduction to Vectors.

Unit 2

Linked Lists: Singly- linked, doubly-linked and circular lists, analysis of insert, delete and search operations in all the three types, implementing sparse matrices. Introduction to Sequences.

Unit 3

Queues: Array and linked representation of queue, de-queue, comparison of the operations on queues in the two representations. Applications of queues.

Unit 4

Stacks: Array and linked representation of stacks, comparison of the operations on stacks in the two representations, implementing multiple stacks in an array; applications of stacks: prefix, infix and postfix expressions, utility and conversion of these expressions from one to another; applications of stacks to recursion: developing recursive solutions to simple problems, advantages and limitations of recursion

Unit 5

Trees and Heaps: Introduction to tree as a data structure; binary trees, binary search trees, analysis of insert, delete, search operations, recursive and iterative traversals on binary search trees. Height-balanced trees (AVL), B trees, analysis of insert, delete, search operations on AVL and B trees.

Introduction to heap as a data structure. analysis of insert, extract-min/max and delete-min/max operations, applications to priority queues.

Unit 6

Hash Tables: Introduction to hashing, hash tables and hashing functions -insertion, resolving collision by open addressing, deletion, searching and their analysis, properties of a good hash function.

Practical

1. Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.
2. WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists (include a function and also overload operator +).
4. Implement Doubly Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
5. Implement Circular Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation. Use Templates.
8. Perform Queues operations using Circular Array implementation. Use Templates.
9. Create and perform different operations on Double-ended Queues using Linked List implementation.

10. WAP to scan a polynomial using linked list and add two polynomial.
11. WAP to calculate factorial and to compute the factors of a given no. (i)using recursion, (ii) using iteration
12. (ii) WAP to display fibonacci series (i)using recursion, (ii) using iteration
13. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion
14. WAP to create a Binary Search Tree and include following operations in tree: (a) Insertion (Recursive and Iterative Implementation) (b) Deletion by copying (c) Deletion by Merging (d) Search a no. in BST (e) Display its preorder, postorder and inorder traversals Recursively (f) Display its preorder, postorder and inorder traversals Iteratively (g) Display its level-by-level traversals (h) Count the non-leaf nodes and leaf nodes (i) Display height of tree (j) Create a mirror image of tree (k) Check whether two BSTs are equal or not
15. WAP to convert the Sparse Matrix into non-zero form and vice-versa.
16. WAP to reverse the order of the elements in the stack using additional stack.
17. WAP to reverse the order of the elements in the stack using additional Queue.
18. WAP to implement Diagonal Matrix using one-dimensional array.
19. WAP to implement Lower Triangular Matrix using one-dimensional array.
20. WAP to implement Upper Triangular Matrix using one-dimensional array.
21. WAP to implement Symmetric Matrix using one-dimensional array.
22. WAP to create a Threaded Binary Tree as per inorder traversal, and implement operations like finding the successor / predecessor of an element, insert an element, inorder traversal.
23. WAP to implement various operations on AVL Tree.
24. WAP to implement heap operations.

References

1. Drozdek, A., (2012), *Data Structures and algorithm in C++*. 3rd edition. Cengage Learning.
2. Goodrich, M., Tamassia, R., & Mount, D., (2011). *Data Structures and Algorithms Analysis in C++*. 2nd edition. Wiley.

Additional Resources

1. Foruzan, B.A. (2012) *Computer Science: A Structured Approach Using C++*, Cengage Learning
2. Lafore, R. (2008). *Object Oriented Programming in C++*. 4th edition. SAMS Publishing.

3. Sahni, S. (2011). *Data Structures, Algorithms and applications in C++*. 2nd Edition, Universities Press
4. Tenenbaum, A. M., Augenstein, M. J., & Langsam Y., (2009), *Data Structures Using C and C++*. 2nd edition. PHI.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1 -2	Single and Multi-dimensional arrays, row and column major –order, static vs. dynamic data structures
3-4	Linked Lists, doubly linked list, circular lists, implementation of link list in array, using pointers, analysis of linked Lists operations, sparse matrices, sequences
5-6	Queues, storage and retrieval operations, implementation of queues,
7-8	Stacks, storage and retrieval operations, implementation of stacks, applications of stacks
9-10	Binary Trees, Recursive and iterative methods of tree traversal
11-13	AVL and B Trees
14	Heaps
15	Hash Tables

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Arrays and linked lists, stacks, queues, tree, heap, hashing, recursion

Operating system (BHCS06) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

The course introduces the students to different types of operating systems. Operating system modules such as memory management, process management and file management are covered in detail.

Course Learning Outcomes

On successful completion of the course, the students will be able to:

1. Implement multiprogramming, multithreading concepts for a small operating system.
2. Create, delete, and synchronize processes for a small operating system.
3. Implement simple memory management techniques.
4. Implement CPU and disk scheduling algorithms.
5. Use services of modern operating system efficiently
6. Implement a basic file system.

Detailed Syllabus

Unit 1

Introduction: Operating systems (OS) definition, Multiprogramming and Time Sharing operating systems, real time OS, Multiprocessor operating systems, Multicore operating systems, Various computing environments.

Unit 2

Operating System Structures: Operating Systems services, System calls and System programs, operating system architecture (Micro Kernel, client server) operating

Unit 3

Process Management: Process concept, Operation on processes, Multi-threaded processes and models, Multicore systems, Process scheduling algorithms, Process synchronization. The Critical-section problem and deadlock characterization, deadlock handling.

Unit 4

Memory Management: Physical and Logical address space; Memory allocation strategies - Fixed and Variable Partitions, Paging, Segmentation, Demand Paging and virtual memory, Page Replacement algorithm.

Unit 5

File and I/O Management: Directory structure, File access methods, Disk scheduling algorithms.

Practical

1. Write a program (using fork() and/or exec() commands) where parent and child execute: a) same program, same code. b) same program, different code. - c) before terminating, the parent waits for the child to finish its task.
2. Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information)
3. Write a program to report behaviour of Linux kernel including information on 19 configured memory, amount of free and used memory. (memory information)
4. Write a program to print file details including owner access permissions, file access time, where file name is given as argument.
5. Write a program to copy files using system calls.
6. Write a program to implement FCFS scheduling algorithm.
7. Write a program to implement Round Robin scheduling algorithm.
8. Write a program to implement SJF scheduling algorithm.
9. Write a program to implement non-preemptive priority based scheduling algorithm.
10. Write a program to implement preemptive priority based scheduling algorithm.
11. Write a program to implement SRJF scheduling algorithm.
12. Write a program to calculate sum of n numbers using thread library.
13. Write a program to implement first-fit, best-fit and worst-fit allocation strategies.

References

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2008). *Operating Systems Concepts*. 8th edition.. John Wiley Publications.

Additional Resources

1. Dhamdhare, D. M. (2006). *Operating Systems: A Concept-based Approach*. 2nd edition. Tata McGraw-Hill Education.

2. Kernighan, B. W., & Rob Pike, R. (1984). *The Unix programming environment* (Vol. 270). Englewood Cliffs, NJ: Prentice-Hall
3. Stallings, W. (2018). *Operating Systems: Internals and Design Principles*. 9th edition. Pearson Education.
4. Tanenbaum, A. S. (2007). *Modern Operating Systems*. 3rd edition. Pearson Education.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Operating System, Definition and its purpose, Time sharing, Multiprogramming and Multiprocessing, Operating System Operations
2	Operating System Services, User and Operating System Interface, System Calls and its Types.
3	Operating system Design and Structure, System Programs, System Boot, Process
4	Operations on Processes, Inter process communication, Shared memory.
5	Multithreading Models, Multicore Programming, Thread Libraries
6	Process Scheduling criteria, Process Scheduling Algorithms, Multiple Processor Scheduling.
7	Process Synchronization, Critical Section Problem, Semaphores.
8	Deadlock Characterization, Methods for handling deadlocks.
9-10	Memory Allocation Strategies-Fixed and Variable partition, Swapping, Logical and Physical Address Space, Paging, Structure of Page Table and its Variations, Shared pages, Segmentation
11-12	Virtual memory, Page Replacement Algorithms, Allocation of frames, Thrashing, Working set model.

13-14	File System , File Characteristics, Access methods, Directory and Disk structure , File system structure and implementation, Directory implementation, Free space Implementation, File Allocation methods.
15	Overview of Secondary Devices, Disk Scheduling Algorithms

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Types of operating systems, memory management, process management, file and I/O management

Computer Networks (BHCS07) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

This course covers the concepts of data communication and computer networks. It comprises of the study of the standard models for the layered protocol architecture to communicate between autonomous computers in a network and also the main features and issues of communication protocols for different layers. Topics covered comprise of introduction to OSI and TCP/IP models also.

Course Learning Outcomes

On successful completion of the course, the student will be able to:

1. Describe the hardware, software components of a network and their interrelations.
2. Compare OSI and TCP/IP network models.
3. Describe, analyze and compare different data link, network, and transport layer protocols.
4. Design/implement data link and network layer protocols in a simulated networking environment.

Detailed Syllabus

Unit 1

Introduction: Types of computer networks, Internet, Intranet, Network topologies, Network classifications.

Unit 2

Network Architecture Models: Layered architecture approach, OSI Reference Model, TCP/IP Reference Model.

Unit 3

Physical Layer: Analog signal, digital signal, digital modulation techniques (ASK, PSK, QAM), encoding techniques, maximum data rate of a channel, transmission media (guided transmission media, wireless transmission, satellite communication), multiplexing (frequency division multiplexing, time division multiplexing, wavelength division multiplexing).

Unit 4

Data Link MAC Layer: Data link layer services, error-detection and correction techniques, error recovery protocols (stop and wait, go back n, selective repeat), multiple access protocols, (TDMA/FDP, CDMA/FDD/CSMA/CD, CSMA/CA), Datalink and MAC addressing, Ethernet, data link layer switching, point-to-point protocol.

Unit 5

Network layer: Networks and Inter networks, virtual circuits and datagrams, addressing, sub netting, Routing- (Distance vector and link state routing), Network Layer Protocols- (ARP, IPV4, ICMP, IPV6).

UNIT 6

Transport and Application Layer: Process to process Delivery- (client server paradigm, connectionless versus connection oriented service, reliable versus unreliable); User Datagram Protocols, TCP/IP protocol, Flow Control.

UNIT 7

Protocols: FTP (File Transfer protocol), SMTP (Simple, Mail Transfer Protocol), Telnet and remote login protocol, WWW (World Wide Web), HTTP (Hyper Text Transfer protocol), Uniform Resource Locator, HTML and forms.

Practical

1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.
2. Simulate and implement stop and wait protocol for noisy channel.

3. Simulate and implement go back n sliding window protocol.
4. Simulate and implement selective repeat sliding window protocol.
5. Simulate and implement distance vector routing algorithm
6. Simulate and implement Dijkstra algorithm for shortest path routing.

References:

1. Forouzan, B. A. (2017). *Data Communication and Networking*. McGraw-Hill Education
2. Tanenbaum, A.S. & Wethrall,D.J. (2012). *Computer Networks*. Pearson Education

Additional References

1. Kozierok, C.M. *The TCP/IP Guide*, free online resource. (2005.). Retrieved from <http://www.tcpipguide.com/free/index.htm>
2. Kurose, J. F., & Ross, K. W. (2017). *Computer Networking: A Top-Down Approach*. Pearson Education India
3. Stallings, W. (2017). *Data and Computer Communications*. 10th edition. Pearson Education India.

Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Topics to be covered
1	Introduction to Computer Networks: Network definition, types of computer networks, Internet, intranet, network topologies, and network classifications.
2	Network Performance issues and concepts: Putting network performance in perspective, balancing network performance with key non-performance characteristics.
3	Performance measurements: speed, bandwidth, throughput and latency; simplex, half duplex and full duplex operation; Quality of service.
4	Network Architecture Models: Layered Approach, OSI Reference Model, TCP/IP Reference Model.

5	Network devices: hubs, switches, bridges, routers, gateways. Physical Layer: Analog signal, digital signal.
6	Physical Layer: digital modulation techniques (ASK, PSK, QAM), encoding techniques, frequency division multiplexing, time division multiplexing.
7	Physical Layer: switching techniques- Circuit, packet and message switching, guided transmission media, wireless transmission, satellite communication Data Link Layer: data link layer services, framing and flow control.
8	Data Link Layer: error-detection and correction techniques error recovery protocols (stop and wait (for noiseless and noisy environment)).
9	Data Link Layer: error recovery protocols (go back n, selective repeat), multiple access protocols, addressing, Ethernet, data link layer switching, point-to-point protocol.
10-11	Network layer: Inter networks, virtual circuits and datagrams, addressing-sub netting, Routing- distance vector and link state routing, Network Layer Protocols- ARP, IPV4, ICMP, IPV6.
12	Transport Layer: Process to process Delivery- client server paradigm, connectionless versus connection oriented service, reliable versus unreliable; user datagram Protocol- well known ports, user datagram.
13	Transport Layer: UDP Operation, use of UDP, TCP/IP protocol - well known ports, TCP Service, features.
14	Transport Layer: TCP connection establishment and release, Flow Control. Application Layer: Domain name space, Distribution of Name space, DNS in the Internet, Resolution.
15	Application Layer: WWW and HTTP, Architecture- Client server model, Uniform Resource Locator, HTTP-Transaction, HTTP operational model and client server communication, HTTP message format.

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

network topologies, OSI model, TCP/IP model, client server model.

Design and Analysis of Algorithms (BHCS08) Discipline Specific Core Course - (DSC) Credit: 06

Course Objective

This course is designed to introduce the students to design and analyse algorithms in terms of efficiency and correctness. The course focuses on highlighting difference between various problem solving techniques for efficient algorithm design.

Course Learning Outcomes:

On successful completion of this course, the student will be able to:

1. Given an algorithm, identify the problem it solves.
2. Write algorithms choosing the best one or a combination of two or more of the algorithm design techniques: Iterative, divide-n-conquer, Greedy, Dynamic Programming using appropriate data structures.
3. Write proofs for correctness of algorithms.
4. Re-write a given algorithm replacing the (algorithm design) technique used with a more appropriate/efficient (algorithm design) technique.

Detailed Syllabus

Unit 1

Algorithm Design Techniques: Iterative technique: Applications to Sorting and Searching (review), their correctness and analysis. Divide and Conquer: Application to Sorting and Searching (review of binary search), merge sort, quick sort, their correctness and analysis.

Dynamic Programming: Application to various problems (for reference; Weighted Interval Scheduling, Sequence Alignment, Knapsack), their correctness and analysis. Greedy Algorithms: Application to various problems, their correctness and analysis.

Unit 2

More on Sorting and Searching: Heapsort, Lower Bounds using decision trees, sorting in Linear Time - Bucket Sort, Radix Sort and Count Sort, Medians & Order Statistics, complexity analysis and their correctness.

Unit 3

Advanced Analysis Technique: Amortized analysis

Unit 4

Graphs: Graph Algorithms - Breadth First Search, Depth First Search and its Applications.

Practical

1. a) Implement Insertion Sort (The program should report the number of comparisons)
b) Implement Merge Sort (The program should report the number of comparisons)
2. Implement Heap Sort (The program should report the number of comparisons)
3. Implement Randomized Quick sort (The program should report the number of comparisons)
4. Implement Radix Sort
5. Create a Red-Black Tree and perform following operations on it: i. Insert a node ii. Delete a node iii. Search for a number & also report the color of the node containing this number.
6. Write a program to determine the LCS of two given sequences
7. Implement Breadth-First Search in a graph
8. Implement Depth-First Search in a graph
9. Write a program to determine the minimum spanning tree of a graph

For the algorithms at S.No 1 to 3 test run the algorithm on 100 different inputs of sizes varying from 30 to 1000. Count the number of comparisons and draw the graph. Compare it with a graph of $n \log n$.

References

1. Kleinberg, J., & Tardos, E. (2013). *Algorithm Design*. 1st edition. Pearson Education India.

Additional Resources

1. Cormen, T.H., Leiserson, C.E. Rivest, R.L., & Stein, C. (2015). *Introduction to Algorithms*. 3rd edition. PHI.
2. Sarabasse & Gleder A. V. (1999). *Computer Algorithm – Introduction to Design and Analysis*. 3rd edition. Pearson Education

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Iterative technique: Applications to Sorting and Searching (review), their correctness and analysis
2	Divide and Conquer: Application to Sorting and Searching (review of binary search), merge sort, their correctness and analysis.
3	Divide and Conquer: quick sort, its correctness and analysis.
4	Heapsort, its correctness and analysis
5	Lower Bounds using decision trees, sorting in Linear Time - Bucket Sort, Radix Sort and Count Sort, their analysis
6	Medians & Order Statistics with analysis
7-9	Graph Algorithms: Graph Representation, Breadth First Search, Depth First Search, Applications
10-11	Greedy Algorithms: Application to various problems, their correctness and analysis
12-14	Dynamic Programming: Application to various problems, their correctness and analysis
15	Amortized analysis

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Brute Force Algorithm, divide and conquer, greedy, dynamic programming approaches, inplace algorithm, best / average / worst case running time of algorithms.

Software Engineering (BHCS09) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

The course introduces fundamental Software Engineering approaches and techniques for software development. The students also develop a case study using appropriate software model.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Analyse and model customer's requirements and model its software design.
2. Use suitable software model for the problem at hand.
3. Estimate cost and efforts required in building software.
4. Analyse and compute impact of various risks involved in software development.
5. Design and build test cases, and to perform software testing.

Detailed Syllabus

Unit 1

Introduction: Software Engineering - A Layered Approach; Software Process – Process Framework, Umbrella Activities; Process Models – Waterfall Model, Incremental Model, and Evolutionary process Model (Prototyping, Spiral Model); Introduction to Agile – Agility Principles, Agile Model – Scrum.

Unit 2

Software Requirements Analysis and Specifications: Use Case Approach, Software Requirement Specification Document, Flow oriented Modeling, Data Flow Modeling, Sequence Diagrams

Unit 3

Design Modeling: Translating the Requirements model into the Design Model, The Design Process, Design Concepts - Abstraction, Modularity and Functional Independence; Architectural Mapping using Data Flow.

Unit 4

Software Metrics and Project Estimations: Function based Metrics, Software Measurement, Metrics for Software Quality; Software Project Estimation (FP based estimations, COCOMO II Model); Project Scheduling (Timeline charts, tracking the schedule).

Unit 5

Quality Control and Risk Management: Quality Control and Quality Assurance, Software Process Assessment and Improvement Capability Maturity Model Integration (CMMI); Software Risks, Risk Identification, Risk Projection and Risk Refinement, Risk Mitigation, Monitoring and Management.

Unit 6

Software Testing: Strategic Approach to Software Testing, Unit Testing, Integration Testing, Validation Testing, System Testing; Black-Box and White Box Testing, Basis Path Testing.

Practical

Practical problems related to

1. Requirement Analysis, Creating a Data Flow, Data Dictionary, Use Cases
3. Computing FP, Effort, Schedule, Risk Table, Timeline chart
4. Design Engineering, Architectural Design, Data Design, Component Level Design
5. Testing, Basis Path Testing

Sample Projects:

1. Criminal Record Management: Implement a criminal record management system for jailers, police officers and CBI officers
2. DTC Route Information: Online information about the bus routes and their frequency and fares
3. Car Pooling: To maintain a web based intranet application that enables the corporate employees within an organization to avail the facility of carpooling effectively.
4. Patient Appointment and Prescription Management System
5. Organized Retail Shopping Management Software
6. Online Hotel Reservation Service System
7. Examination and Result computation system
8. Automatic Internal Assessment System
9. Parking Allocation System
10. Wholesale Management System

References

1. Aggarwal, K. K., & Singh, Y. (2007). *Software Engineering*. 3rd edition. New Age International Publishers.
2. Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach*. 8th edition. McGraw-Hill.

Additional Resources

1. Jalote, P. (2005). *An Integrated Approach to Software Engineering*. 3rd edition. Narosa Publishing House.
2. Schwaber, K. & Sutherland, J. (2016). *The Definitive Guide to Scrum: The Rules of the Game*. [<https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf>]
3. Sommerville. (2011). *Software Engineering*. 9th edition. Addison Wesley.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class-room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Software - Nature of Software, Software Application Domains, Legacy Software; Software Engineering - A Layered Approach; Software Process – Process Framework, Framework and Umbrella Activities
2	Process Models – Waterfall Model, Incremental Model, and Evolutionary process Model (Prototyping, Spiral Model);
3	Introduction to Agile – Agility, Cost of Change, Agility Principles
4	Agile Model - Scrum; Software Process Assessment and Improvement - Capability Maturity Model Integration (CMMI).
5	Requirements Modeling - Requirements Modeling Approaches, Flow oriented

	Modeling, Data Flow Modeling,
6	Control Flow Model, Control Specification, Process Specification, Behavioral Model, State Diagram, Sequence Diagrams;
7	Design Modeling - Design Concepts, Translating requirements model into design model, Design Process, Abstraction, Architecture, Separation of concerns, Modularity, Information hiding, Functional Independence,
8	Refinement, Refactoring; Architectural Mapping using Data Flow.
9	Risk Management- Software Risks, Risk Identification, Risk Projection and Risk Refinement, Risk Mitigation, Monitoring and Management.
10	Function based Product Metrics, Software Quality Metrics;
11	Estimation for Software Project, Project Scheduling, Quality - Software Quality, McCall's Quality Factors, ISO 9126 Quality Factors, Achieving Software Quality;
12	Cost Impact of Software Defects, Defect Amplification and Removal, Formal Technical Reviews; Software Quality Assurance – SQA Tasks.
13-14	Software Testing - Strategic Approach to Software Testing, Unit Testing, Integration Testing, Validation Testing, System Testing;
15	Black-Box and White Box Testing, Basis Path Testing

Assessment Methods

Written tests, assignments, quizzes, presentations, projects as announced by the instructor in the class.

Keywords

Software models, requirement analysis, software design and testing, software risks and costs

Database Management Systems (BHCS10) Discipline Specific Core Course - (DSC) Credit: 06

Course Objective

The course introduces the foundations of database management systems focusing on significance of a database, relational data model, schema creation and normalization, transaction processing, indexing, and the relevant data structures (files and B+-trees).

Course Learning Outcomes

On successful completion of the course, students will:

1. Describe major components of DBMS and their functions
2. Model an application's data requirements using conceptual modelling tools like ER diagrams and design database schemas based on the conceptual model.
3. Write queries in relational algebra / SQL
4. Normalize a given database schema to avoid data anomalies and data redundancy.
5. Describe the notions of indexes, views, constraints and transactions.

Detailed Syllabus

Unit 1

Introduction to databases: Characteristics of database approach, data models, database system architecture, data independence and data abstraction.

Unit 2

Data modeling: Entity relationship (ER) modeling: Entity types, relationships, constraints, ER diagrams, EER model

Unit 3

Relation data model: Relational model concepts, relational constraints, relational algebra.

Unit 4

SQL queries: SQL data definition, data types, specifying constraints, Queries for retrieval, insertion, deletion, updation, introduction to views.

Unit 5

Database design: Mapping ER/EER model to relational database, functional dependencies, Lossless decomposition, Normal forms (upto BCNF).

Unit 6

Transaction and data storage: Introduction to transaction processing: ACID properties, concurrency control; Introduction to indexing structures for files.

Practical

Create and use the following database schema to answer the given queries.

EMPLOYEE Schema

Field Type NULL KEY

DEFAULT

Eno Char(3) NO PRI NIL

Ename Varchar(50) NO NIL

Job_type Varchar(50) NO NIL

Manager Char(3) Yes FK NIL

Hire_date Date NO NIL

Dno Integer YES FK NIL

Commission Decimal(10,2) YES NIL

Salary Decimal(7,2) NO NIL

DEPARTMENT Schema

Field Type NULL KEY

DEFAULT

Dno Integer No PRI NULL

Dname Varchar(50) Yes NULL

Location Varchar(50) Yes New Delhi

Query List

1. Query to display Employee Name, Job, Hire Date, Employee Number; for each employee with the Employee Number appearing first.

2. Query to display unique Jobs from the Employee Table.
3. Query to display the Employee Name concatenated by a Job separated by a comma.
4. Query to display all the data from the Employee Table. Separate each Column by a comma and name the said column as THE_OUTPUT.
5. Query to display the Employee Name and Salary of all the employees earning more than \$2850.
6. Query to display Employee Name and Department Number for the Employee No= 7900.
7. Query to display Employee Name and Salary for all employees whose salary is not in the range of \$1500 and \$2850.
8. Query to display Employee Name and Department No. of all the employees in Dept 10 and Dept 30 in the alphabetical order by name.
9. Query to display Name and Hire Date of every Employee who was hired in 1981.
10. Query to display Name and Job of all employees who don't have a current Manager.
11. Query to display the Name, Salary and Commission for all the employees who earn commission.
12. Sort the data in descending order of Salary and Commission.
13. Query to display Name of all the employees where the third letter of their name is 'A'.
14. Query to display Name of all employees either have two 'R's or have two 'A's in their name and are either in Dept No = 30 or their Manger's Employee No = 7788.
15. Query to display Name, Salary and Commission for all employees whose Commission amount is 14 greater than their Salary increased by 5%.
16. Query to display the Current Date.
17. Query to display Name, Hire Date and Salary Review Date which is the 1st Monday after six months of employment.
18. Query to display Name and calculate the number of months between today and the date each employee was hired.

19. Query to display the following for each employee <E-Name> earns < Salary> monthly but wants < 3 * Current Salary >. Label the Column as Dream Salary.
20. Query to display Name with the 1st letter capitalized and all other letter lower case and length of their name of all the employees whose name starts with 'J', 'A' and 'M'.
21. Query to display Name, Hire Date and Day of the week on which the employee started.
22. Query to display Name, Department Name and Department No for all the employees.
23. Query to display Unique Listing of all Jobs that are in Department # 30.
24. Query to display Name, Dept Name of all employees who have an 'A' in their name.
25. Query to display Name, Job, Department No. And Department Name for all the employees working at the Dallas location.
26. Query to display Name and Employee no. Along with their Manger's Name and the Manager's employee no; along with the Employees' Name who do not have a Manager.
27. Query to display Name, Dept No. And Salary of any employee whose department No. and salary matches both the department no. And the salary of any employee who earns a commission.
28. Query to display Name and Salaries represented by asterisks, where each asterisk (*) signifies \$100.
29. Query to display the Highest, Lowest, Sum and Average Salaries of all the employees
30. Query to display the number of employees performing the same Job type functions.
31. Query to display the no. of managers without listing their names.
32. Query to display the Department Name, Location Name, No. of Employees and the average salary for all employees in that department.
33. Query to display Name and Hire Date for all employees in the same dept. as Blake.
34. Query to display the Employee No. And Name for all employees who earn more than the average salary.
35. Query to display Employee Number and Name for all employees who work in a department with any employee whose name contains a 'T'.

36. Query to display the names and salaries of all employees who report to King.
37. Query to display the department no, name and job for all employees in the Sales department

References

1. Elmasri, R., & Navathe, S.B. (2015). *Fundamentals of Database Systems*. 7th edition. Pearson Education.

Additional Resources

1. Date, C. J. (2004). *An Introduction to database systems*. 8th edition. Pearson Education.
2. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). *Database System Concepts*. 6th edition. McGrawHill.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Introduction to databases: Characteristics of database approach, data models, database system architecture, data independence and data abstraction.
2-3	Entity relationship(ER) modeling: Entity types, relationships, constraints;ER examples
4-5	Relation data model: Relational model concepts, relational constraints, relational algebra; examples
6-8	SQL queries; examples
9	Database design: Mapping ER/EER model to relational database; examples
10-12	Database design: functional dependencies, Lossless decomposition, Normal forms (upto BCNF); examples
13	Transaction and data storage: Transaction processing:ACID properties, concurrency control; File structure and indexing: Operations on files, File of Unordered and ordered records

14	File structure and indexing: overview of File organizations, Indexing structures for files, examples
15	XML databases, noSQL systems

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Entity-Relationship Modeling, Database Design, Transaction Processing, noSQL systems.

Internet Technologies (BHCS11) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

This course introduces the protocols used in Internet, its architecture, and security aspect of Internet. Student will have an insight that how a search engine works and web crawls.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Describe Internet, its architecture, services and protocol.
2. Implement a simple search engine.
3. Implement a web crawler.
4. Use javascript technologies to make a website highly responsive, more efficient and user friendly

Detailed Syllabus

Unit 1

Introduction: Network address translation, Subnet Masking, Difference between Intranet and Internet, Working of Internet, Dynamic and Static Routing, Domain Name Server , networking tools - ipconfig, ping, netstat, traceroute

Unit 2

Introduction to Internet Protocols: HTTP, HTTPS, FTP, SMTP, IMAP, POP3, VoIP

Unit 3

Web Servers: Introduction, Working, Configuring, Hosting and Managing a Web server,

Proxy Servers: Introduction, Working, Type of Proxies, setting up and managing a proxy server

Client-side Technologies, Server-side Technologies and hybrid technologies

Unit 4

Javascript, jQuery, JSON, NODE.js, BOOTSTRAP, Introduction to forums, blogging, portfolio,

developing a responsive website, Combining Web Applications and Mobile Applications

Unit 5

Search Engines - components, working, optimization, Crawling, BOTS

Unit 6

Introduction to cookies and sessions, Introduction to e-commerce websites and e-carts.

Practical

Pre-requisites for course: Programming, Computer Networks, Web-Designing (HTML, CSS, Basic JavaScript)

1. Demonstrate the use of networking tools like ping, ipconfig, netstat and traceroute.
2. Configure a web-server on a personal system.
3. Demonstrate the network monitoring of the internet traffic through any predefined tool
4. Develop an interactive website using jquery, JSON, NODE.js and BOOTSTRAP with following functionalities.
 1. Design a home page and other allied pages of the website using HTML and CSS
 2. Create a registration form and insert the data into tables at the backend. Creating an html form with content validation using JavaScript.
 3. Handle HTML form using jQuery, store the data in JSON objects, pass them to another page and display it there using jQuery
 4. Logging system to manage various types of accounts
 5. Create pages with dynamic content fetching and display
 6. Perform event handling in node.js

References

1. Bayross, I. (2013). *Web enabled commercial application development using HTML, JavaScript, DHTML and PHP*. 4th edition. BPB Publication.
2. DComer. (2018). *The Internet Book: Everything You need to know about Computer networking and how the internet works*. 5th edition. CRC Press.
3. Duckett, J.(2014). *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley

Additional Resources

1. Godbole, A. S.& Kahate A (2008). *Web Technologies*. Tata McGrawHill
2. Greenlaw R. & Hepp E, (2007). *Fundamentals of Internet and WWW*. 2nd edition. Tata McGrawHill.
3. Jackson. (2008). *Web Technologies*. Pearson Education
4. Patel, B & Barik, L.B , *Internet & Web Technology* , Acme Learning Publisher.
5. Reddy, S., Aggarwal, A., Sayer, M., Totty, B., & Gourley, D. (2002). *HTTP: The Definitive Guide*. Media: O'Reilly Media Inc.
6. Young, M. L. (2007). *The Complete reference to Internet*. Tata: McGraw Hill.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class-room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-2	Network address translation, Subnet Masking, Difference between Intranet and Internet, Working of Internet, Dynamic and Static Routing, Domain Name Server, networking tools - ipconfig, ping, netstat, traceroute
3	Introduction to Internet Protocols - HTTP, HTTPS, FTP, SMTP, IMAP, POP3, VoIP
4-7	Web Servers: Working, Configuring, Hosting and Managing a Web server Proxy Servers: Working, Type of Proxies, setting up and managing a

	proxy server, Client-side Technologies, Server-side Technologies and hybrid technologies
8-10	javascript, JSON jQuery
11-12	NODE.js, BOOTSTRAP
13-14	Introduction to forums, blogging, portfolio, Developing a responsive website, combining Web Applications and Mobile Applications
15	Search Engines - components, working, optimization, Crawling, BOTS Introduction to cookies and sessions, e-commerce websites and e-carts

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Internet, networks, JSON, AJAX, JQUERY, web application

Theory of Computation (BHCS12) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

This course introduces formal models of computation, namely, finite automaton, pushdown automaton, and Turing machine; and their relationships with formal languages. Students will also learn about the limitations of computing machines.

Course Learning Outcomes

On successful completion of the course, a student will be able to:

1. Design a finite automaton, pushdown automaton or a Turing machine for a problem at hand.
2. Apply pumping lemma to prove that a language is non-regular/non-context-free.
3. Describe limitations of a computing machine.

Detailed Syllabus

Unit 1

Languages: Alphabets, string, language, basic operations on language, concatenation, union, Kleene star.

Unit 2

Regular Expressions and Finite Automata: Regular expressions, Deterministic finite automata (DFA).

Unit 3

Regular Languages: Non-deterministic Finite Automata (NFA), relationship between NFA and DFA, Transition Graphs (TG), properties of regular languages, the relationship between regular languages and finite automata, Kleene's Theorem.

Unit 4

Non-Regular Languages and Context Free Grammars: Pumping lemma for regular grammars, Context-Free Grammars (CFG),

Unit 5

Context-Free Languages (CFL) and PDA: Deterministic and non-deterministic Pushdown Automata (PDA), parse trees, leftmost derivation, pumping lemma for CFL, properties of CFL.

Unit 6

Turing Machines and Models of Computations: Turing machine as a model of computation, configuration of simple Turing machine, Church Turing Thesis, Universal Turing Machine, decidability, halting problem.

Practical

Tutorials based on theory.

References

1. Cohen, D. I. A. (2011). *Introduction to Computer Theory*. 2nd edition. Wiley India.
2. Lewis, H.R. & Papadimitriou, H. R. (2002). *Elements of the Theory of Computation*. 6th edition. Prentice Hall of India (PHI)

Additional Resources

1. Goodrich, M., Tamassia, R., & Mount, D.M. (2011). *Data Structures and Algorithms Analysis in C++*. 2nd edition. Wiley.
2. Gopalkrishnan, G.L. (2019) *Automata and Computability: A programmer's perspective*. CRC Press.

3. Linz, P. (2016). *An Introduction to Formal Languages and Automata*. 6th edition. Jones and Bartlett Learning.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Topics to be covered
1	Languages: Alphabets, string, language, Basic operations on a Language, concatenation, Kleene Star, Kleene closure.
2	Regular Expression: Definition and use of regular expressions, languages defined by regular expressions, understanding a regular expression, building regular expressions
3	Introduction to finite automata and its relationship with regular expressions, Finite Automata and their languages, deterministic finite automata (DFA).
4	Transition Graphs Relaxing Restrictions on Inputs in TG (Transition Graph), TG vs. FA, Generalized Transition Graphs (GTG), Introduction to Non-determinism.
5	Kleene's Theorem: Turning TGs and FA to regular expressions and vice versa, Depicting union of two Regular Languages (RL) using an FA, Depicting concatenation (Product) of two RL using an FA.
6	Keene Star of a RL (Regular Language) using an FA, Non-deterministic finite automata (NFA), relationship between NFA and DFA, converting NFA to DFA.
7	Regular Languages: Complement and intersection of a regular languages, relationship between regular languages and finite automata.
8	Pumping lemma for regular languages. Introduction to context-free languages.
9	Context Free Grammar: Context free grammars, Parse trees. Introduction to Pushdown Automata (PDA). Pushdown Automata: A new Format for FAs, Introduction to Pushdown Automata (PDA).
10	Pushdown Automata: Adding a pushdown stack, design and analysis of Deterministic PDA, design and analysis of non-deterministic pushdown automata.
11	Non-Context-Free Languages: Pumping Lemma for Context-Free-Languages (CFLs), properties of context free languages.

12	Simple Turing machine as a model of computation and its configuration, computing with Turing machine and its working.
13	Building simple Turing machines, combining Turing machines, Church Turing Thesis.
14-15	Universal Turing machine, semi-decidability and decidability, recursively-enumerable and recursive languages, halting problem.

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Regular expressions and languages, finite automata, context free grammar and languages, pushdown automata, Turing machine.

Artificial Intelligence (BHCS13) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

This course introduces the basic concepts and techniques of Artificial Intelligence (AI). The course aims to introduce intelligent agents and reasoning, heuristic search techniques, game playing, knowledge representation, reasoning with uncertain knowledge.

Course Learning Outcomes

On successful completion of this course, students will be able to:

1. Identify problems that are amenable to solution by specific AI methods
2. Represent knowledge in Prolog and write code for drawing inferences.
3. Identify appropriate AI technique for the problem at hand
4. Compare strengths and weaknesses of different artificial Intelligence techniques.
5. Sensitive towards development of responsible Artificial Intelligence

Detailed Syllabus

Unit 1

Introduction: Introduction to artificial intelligence, background and applications, Turing test, rational agents, intelligent agents, structure, behaviour and environment of intelligent agents.

Unit 2

Knowledge Representation: Propositional logic, first order predicate logic, resolution principle, unification, semantic nets, conceptual dependencies, frames, scripts, production rules, conceptual graphs.

Unit 3

Reasoning with Uncertain Knowledge: Uncertainty, non-monotonic reasoning, truth maintenance systems, default reasoning and closed world assumption, Introduction to probabilistic reasoning, Bayesian probabilistic inference, introduction to fuzzy sets and fuzzy logic, reasoning using fuzzy logic.

Unit 4

Problem Solving and Searching Techniques: Problem characteristics, production systems, control strategies, breadth first search, depth first search, hill climbing and its variations, heuristics search techniques: best first search, A* algorithm, constraint satisfaction problem, means-end analysis.

Unit 5

Game Playing: introduction to game playing, min-max and alpha-beta pruning algorithms.

Prolog Programming: Introduction to Programming in Logic (PROLOG), Lists, Operators, basic Input and Output.

Unit 6

Understanding Natural Languages: Overview of linguistics, Chomsky hierarchy of grammars, parsing techniques.

Unit 7

Ethics in AI, Fairness in AI, Legal perspective

Practical

1. Write a prolog program to calculate the sum of two numbers.
2. Write a Prolog program to implement $\max(X, Y, M)$ so that M is the maximum of two numbers X and Y.
3. Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.

4. Write a program in PROLOG to implement generate_fib(N,T) where T represents the Nth term of the fibonacci series.
5. Write a Prolog program to implement GCD of two numbers.
6. Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.
7. Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.
8. Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.
9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.
10. Write a Prolog program to implement reverse (L, R) where List L is original and List R is reversed list.
11. Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.
12. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.
13. Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.
14. Write a Prolog program to implement nth_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.
15. Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.
16. Write a prolog program to implement insert_nth (I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.
17. Write a Prolog program to implement delete_nth (N, L, R) that removes the element on Nth position from a list L to generate a list R.
18. Write a program in PROLOG to implement merge (L1, L2, L3) where L1 is first ordered list and L2 is second ordered list and L3 represents the merged list.

References

1. Rich, E. & Knight, K. (2012). *Artificial Intelligence*. 3rd edition. Tata McGraw Hill.
2. Russell, S.J. & Norvig, P. (2015) *Artificial Intelligence - A Modern Approach*. 3rd edition. Pearson Education

Additional Resources:

1. Bratko, I. (2011). *Prolog Programming for Artificial Intelligence*. 4th edition. Pearson Education
2. Clocksin, W.F. & Mellish (2003), *Programming in PROLOG*. 5th edition. Springer
3. Kaushik, S. (2011). *Artificial Intelligence*. Cengage Learning India.
4. Patterson, D.W. (2015). *Introduction to Artificial Intelligence and Expert Systems*. 1st edition. Pearson Education.

Web Resources

1. <https://cyber.harvard.edu/topics/ethics-and-governance-ai>
2. <https://royalsocietypublishing.org/doi/full/10.1098/rsta.2018.0085>
3. <https://arxiv.org/abs/1812.02953>

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Introduction to artificial intelligence, background and applications, Turing test and rational agent approaches to AI, introduction to intelligent agents.
2	Structure, behavior and environment of intelligent agents, problem characteristics, production systems, control strategies.
3	Introduction to programming in logic (PROLOG).
4	Programming in logic (PROLOG), breadth first search, depth first search introduction of heuristic search techniques.
5	Propositional logic, first order predicate logic.
6	Unification, clausal form, resolution principle.
7	Semantic nets, conceptual graphs, conceptual dependencies.
8	Frames, scripts, Uncertainty: non-monotonic reasoning, truth maintenance systems, default reasoning and closed world assumption.

9	Bayesian probabilistic inference, Bayesian networks, Dempster-Shafer theory, Introduction to fuzzy sets and fuzzy logic.
10	Basic reasoning using fuzzy concepts, production rules, Chomsky hierarchy of grammars, context-free grammars.
11	Hill climbing and its variations, best first search.
12	A* algorithm, constraint satisfaction problem, means-end analysis.
13	Introduction to game playing, min-max procedure, alpha-beta pruning.
14-15	Overview of linguistics, Chomsky hierarchy of grammars, parsing techniques..

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Artificial Intelligence, Problem Solving, Knowledge Representation, Reasoning, Uncertainty, Natural Language Processing

Computer Graphics (BHCS14) Discipline Specific Core Course - (DSC)

Credit: 06

Course Objective

This course introduces fundamental concepts of Computer Graphics with focus on modelling, rendering and interaction aspects of computer graphics. The course emphasizes the basic principles needed to design, use and understand computer graphics system.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Describe Standard raster and vector scan devices as well as Graphical Input and output devices
2. Implement algorithms for drawing basic primitives such as line, circle and ellipse.
3. Implement algorithms for line clipping and polygon clipping and filling.

4. Implement a 3D object representation scheme and carryout 2D and 3D Transformation, 3D projections
5. Implement visible surface determination algorithms, Illumination models and surface rendering methods, color models
6. Implement a simple computer animation algorithm

Detailed Syllabus

Unit 1

Introduction: Introduction to Graphics systems, Basic elements of Computer graphics, Applications of computer graphics. Architecture of Raster and Random scan display devices, input/output devices.

Unit 2

Drawing and clipping primitives: Raster scan line, circle and ellipse drawing algorithms, Polygon filling, line clipping and polygon clipping algorithms

Unit 3

Transformation and Viewing: 2D and 3D Geometric Transformations, 2D and 3D Viewing Transformations (Projections- Parallel and Perspective), Vanishing points.

Unit 4

Geometric Modeling: Polygon Mesh Representation, Cubic Polynomial curves (Hermite and Bezier).

Unit 5

Visible Surface determination and Surface Rendering: Z-buffer algorithm, List-priority algorithm and area subdivision algorithm for visible surface determination. Illumination and shading models, RGB color model and Basics of Computer Animation.

Practical

1. Write a program to implement Bresenham's line drawing algorithm.
2. Write a program to implement mid-point circle drawing algorithm.
3. Write a program to clip a line using Cohen and Sutherland line clipping algorithm.
4. Write a program to clip a polygon using Sutherland Hodgeman algorithm.
5. Write a program to fill a polygon using Scan line fill algorithm.
6. Write a program to apply various 2D transformations on a 2D object (use homogenous

Coordinates).

7. Write a program to apply various 3D transformations on a 3D object and then apply parallel and perspective projection on it.

8. Write a program to draw Hermite /Bezier curve.

References

1. Baker, D.H. (2008). *Computer Graphics*. 2nd edition. Prentice Hall of India.
2. Foley, J. D., Dam, A.V, Feiner, S. K., & Hughes, J. F. (1995). *Computer Graphics: Principles and Practice in C*. 2nd edition. Addison-Wesley Professional.

Additional Resources:

1. Bhattacharya, S. (2018). *Computer Graphics*. Oxford University Press
2. Cohen, D. I. A. (2011). *Introduction to Computer Theory*. 2nd edition. Wiley India.
3. Marschner, S., & Shirley, P. (2017) *Fundamentals of Computer Graphics*. 4th edition. CRC Press
4. Rogers, D. F. (1989). *Mathematical Elements for Computer Graphics*. 2nd edition. McGraw Hill.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Contents
1	Introduction to Graphics systems, Basic elements of Computer graphics, Applications of computer graphics.
2	Graphics Hardware: Architecture of Raster and Random scan display devices, input/output devices.
3-4	Drawing Primitives: Raster scan line drawing algorithm, circle and ellipse drawing algorithms
5	Polygon filling, line clipping and polygon clipping algorithms
6	Transformation: 2D and 3D Geometric Transformations
7-9	Viewing : 3D Viewing Transformations, Parallel Projections, Perspective Projections ,Vanishing points
10	Geometric Modeling: Representing curves(Hermite and Bezier)
11-12	Geometric Modeling: Representing curves(Hermite and Bezier)(cont.), Visible Surface determination: Z-buffer algorithm
13	List-priority algorithm and area subdivision algorithm.
14	Surface rendering: Illumination and shading models
15	RGB color model and Computer Animation.

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Computer Graphics, Modelling, Rendering, Transformation and viewing

Data Analysis and Visualization (BHCS15A) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

This course introduces students to data analysis and visualization in the field of exploratory data science using Python.

Course Learning Outcomes

On successful completion of the course, the students will be able to :

1. Use data analysis tools in the pandas library.
2. Load, clean, transform, merge and reshape data.
3. Create informative visualization and summarize data sets.
4. Analyze and manipulate time series data.
5. Solve real world data analysis problems.

Detailed Syllabus

Unit 1

Introduction: Introduction to Data Science, Exploratory Data Analysis and Data Science Process. Motivation for using Python for Data Analysis, Introduction of Python shell iPython and Jupyter Notebook.

Essential Python Libraries: NumPy, pandas, matplotlib, SciPy, scikit-learn, statsmodels

Unit 2

Getting Started with Pandas: Arrays and vectorized computation, Introduction to pandas Data Structures, Essential Functionality, Summarizing and Computing Descriptive Statistics.

Data Loading, Storage and File Formats.

Reading and Writing Data in Text Format, Web Scraping, Binary Data Formats, Interacting with Web APIs, Interacting with Databases

Data Cleaning and Preparation.

Handling Missing Data, Data Transformation, String Manipulation

Unit 3

Data Wrangling: Hierarchical Indexing, Combining and Merging Data Sets Reshaping and Pivoting.

Data Visualization matplotlib: Basics of matplotlib, plotting with pandas and seaborn, other python visualization tools

Unit 4

Data Aggregation and Group operations: Group by Mechanics, Data aggregation, General split-apply-combine, Pivot tables and cross tabulation

Time Series Data Analysis: Date and Time Data Types and Tools, Time series Basics, date Ranges, Frequencies and Shifting, Time Zone Handling, Periods and Periods Arithmetic, Resampling and Frequency conversion, Moving Window Functions.

Unit 5

Advanced Pandas: Categorical Data, Advanced GroupBy Use, Techniques for Method Chaining

Practical

Use data set of your choice from Open Data Portal (<https://data.gov.in/>) for the following exercises.

1. Practicals based on NumPy ndarray
2. Practicals based on Pandas Data Structures
3. Practicals based on Data Loading, Storage and File Formats
4. Practicals based on Interacting with Web APIs
5. Practicals based on Data Cleaning and Preparation
6. Practicals based on Data Wrangling
7. Practicals based on Data Visualization using matplotlib
8. Practicals based on Data Aggregation
9. Practicals based on Time Series Data Analysis

References

1. McKinney, W.(2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy and IPython*. 2nd edition. O'Reilly Media.
2. O'Neil, C., & Schutt, R. (2013). *Doing Data Science: Straight Talk from the Frontline* O'Reilly Media.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Introduction: What is Data Science? Exploratory Data Analysis and Data Science Process. Why Python for Data Analysis? Introduction of Python shell iPython and

	Jupyter Notebook.
2-3	Essential Python Libraries: Learn NumPy, pandas, matplotlib, SciPy, scikit-learn, statsmodels.
4	Built-in Data Structures, Function and Files: Data Structure and sequences, Functions, Files and Operating systems
5	Arrays and Vectorized computation: The NumPy ndarray, Universal Functions, Array Oriented Programming with Arrays, File Input and Output with Arrays, Linear Algebra, Pseudorandom Number Generation
6	Getting Started with pandas: Introduction to pandas Data Structures, Essential Functionality, Summarizing and Computing Descriptive Statistics.
7	Data Loading, Storage and File Formats: Reading and Writing Data in Text Format, Web Scraping, Binary Data Formats, Interacting with Web APIs, Interacting with Databases.
8	Data Cleaning and Preparation: Handling Missing Data, Data Transformation, String Manipulation
9	Data Wrangling: Hierarchical Indexing, Combining and Merging Data Sets Reshaping and Pivoting.
10	Data Visualization matplotlib: Basics of matplotlib, plotting with pandas and seaborn, other python visualization tools.
11	Data Aggregation and Group operations: Group by Mechanics, Data aggregation, General split-apply-combine, Pivot tables and cross tabulation
12-13	Time Series Data Analysis: Date and Time Data Types and Tools, Time series Basics, date Ranges, Frequencies and Shifting, Time Zone Handling, Periods and Periods Arithmetic, Resampling and Frequency conversion, Moving Window Functions
14-15	Data Analysis Case Studies

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Data Analysis, data wrangling, data visualization, data cleaning, data preparation

System Programming (BHCS15B) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

The course is focused on design of assembler and basic compiler. The course covers topics like absolute loader, relocating loader and dynamic linking.

Course Learning Outcomes

On successful completion of the course, the students will be able to:

1. Describe the working of assemblers and compilers.
2. Use Lex/ Yacc for building basic compiler.
3. Develop a two pass Assemblers.
4. Describe the role of the loaders, linkers and relocatable programs.

Detailed Syllabus

Unit 1

Assemblers & Loaders, Linkers: One pass and two pass assembler, design of an assembler, Absolute loader, relocation and linking concepts, relocating loader and Dynamic Linking.

Unit 2

Introduction: Overview of compilation, Phases of a compiler.

Unit 3

Lexical Analysis: Role of a Lexical analyzer, Specification and recognition of tokens, Symbol table, lexical Analyzer Generator.

Unit 4

Parsing & Intermediate representations: Bottom up parsing- LR parser, yacc, three address code generation, syntax directed translation, translation of types, control statements

Unit 5

Storage organization & Code generation: Activation records, stack allocation, Object code generation

Practical

Projects to implement an assembler for a hypothetical language.

Programs to get familiar with Lex and Yacc

1. Write a Lex program to count the number of lines and characters in the input file.
2. Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in in alphabetical order, wrapping around at Z. e.g. a is replaced by d, b by e, and so on z by c.
3. Write a Lex program that finds the longest word (defined as a contiguous string of upper and lower case letters) in the input.
4. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.
5. Write a Lex program to count the number of identifiers in a C file.
6. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.
7. Write a Lex specification program that generates a C program which takes a string "abcd" and prints the following output
abcd
abc
a
8. A program in Lex to recognize a valid arithmetic expression.
9. Write a YACC program to find the validity of a given expression (for operators + - * and /)A program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lowercase only.
10. A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).
11. Program in YACC to recognize the string „abbb“, „ab“ „a“ of the langauge (an b n , n>=1).
12. Program in YACC to recognize the language (an b , n>=10). (output to say input is valid or not)

References

1. Aho, A., Lam, M., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools*. 2nd edition. Addison Wesley.
2. Chattopadhyaya, S. (2011). *System Software*. P H I Learning.

Additional references:

1. Beck, L. & Manjula, D. (1996). *System Software: An Introduction to System Programming*. 3rd edition. Pearson Education.
2. Dhamdhare, D. M. (2015). *Systems Programming*. Tata McGrawHill.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class-room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-3	Assemblers & Loaders, Linkers: One pass and two pass assembler, design of an assembler, Absolute loader, relocation and linking concepts, relocating loader and Dynamic Linking.
4	Overview of compilation, Phases of a compiler.
5-6	Lexical Analysis: Role of a Lexical analyzer, Specification and recognition of tokens, Symbol table, lexical Analyzer Generator.
7-9	Parsing : Bottom up parsing- LR parser, yacc.
10-11	Intermediate representations: Three address code generation, syntax directed translation, translation of types, control statements
12-15	Storage organization & Code generation: Activation records, stack allocation, Object code generation

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Compilers, lexical analyzer, syntax directed translation, assembler, loader, linker.

Combinatorial Optimization (BHCS15C) Discipline Specific Elective - (DSE)

Credit: 06

Course Objectives

This course is designed to introduce the fundamentals of combinatorial optimization to the students in terms of both theory and applications, so as to equip them to explore the more advanced areas of convex and non-convex optimizations.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Model problems using linear and integer programs
2. Apply polyhedral analysis to develop algorithms for optimization problems
3. Use the concept of duality for design of algorithms

Detailed Syllabus

Unit 1

Introduction to Combinatorial Optimization Problems, Linear and Integer Programs: LP Formulation, understanding integer programs, computational complexities of IP vs LP, using LP to find optimal or approximate integral solutions, concept of integrality gap.

Unit 2

Theory of Linear Programming and Algorithmic Perspective to Simplex Method: standard vs. equational form, basic feasible solutions, convexity and convex polyhedra, correspondence between vertices and basic feasible solutions, geometry of Simplex algorithm, exception handling (unboundedness, degeneracy, infeasibility), Simplex algorithm, avoiding cycles.

Unit 3

Primal-Dual Algorithms: interpretation of dual, optimality conditions for primal and dual, weak and strong duality, complementary slackness, primal-dual algorithm for the shortest path problem.

Unit 4

Network Flows: linear programming formulations for network flows and bipartite matching, totally unimodular matrices integral polyhedral.

Tutorials

Tutorials based on Theory

References

1. Matousek & Gartner (2007). *Understanding and Using Linear Programming*. Springer.
2. Papadimitriou, C.H. & Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and complexity*. Dover Publications.

Additional Resources:

1. Bazaraa, M.S., Jarvis, J.J., & and Sherali, H.D.(2008). *Linear Programming and Network Flows*. 2nd edition. Wiley.
2. Korte, B., & Vygen, J. (2006). *Combinatorial Optimization*. 5th edition. Springer.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-2	Introduction to Combinatorial Optimization Problems, Linear and Integer Programs: LP Formulation, understanding integer programs, computational complexities of IP vs LP, using LP to find optimal or approximate integral solutions, concept of integrality gap
3-6	Theory of Linear Programming and Algorithmic Perspective to Simplex Method: standard vs. equational form, basic feasible solutions, convexity and convex polyhedra, correspondence between vertices and basic feasible solutions, geometry of Simplex algorithm, exception handling (unboundedness, degeneracy, infeasibility), Simplex algorithm, avoiding cyc
7-10	Primal-Dual Algorithms: interpretation of dual, optimality conditions for primal and dual, weak and strong duality, complementary

	slackness, primal-dual algorithm for the shortest path problem.
11-15	Network Flows: linear programming formulations for network flows and bipartite matching, totally uni-modular matrices integral polyhedral

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

optimization problems, linear programming, integer programming, duality, network flow problems

Digital Image Processing (BHCS16A) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

This course introduces students to the fundamentals of digital image processing, and various image transforms, image restoration techniques, image compression and segmentation used in digital image processing.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Describe the roles of image processing systems in a variety of applications;
2. Write programs to read/write and manipulate images: enhancement, segmentation, and compression, spatial filtering.
3. Develop Fourier transform for image processing in frequency domain.
4. Evaluate the methodologies for image segmentation, restoration

Detailed Syllabus

Unit 1

Introduction: Digital Image Fundamentals: Brightness, Adaptation and Discrimination, Light and Electromagnetic Spectrum, Image Sampling and Quantization, Some Basic Relationships between Pixels Types of images.

Unit 2

Spatial Domain Filtering: Some Basic Intensity Transformation Functions, Histogram Equalization, Spatial Correlation and Convolution, Smoothing Spatial Filters: Low pass filters, Order Statistics filters; Sharpening Spatial Filters: Laplacian filter

Unit 3

Filtering in Frequency Domain: The Discrete Fourier Transformation (DFT), Frequency Domain Filtering: Ideal and Butterworth Low pass and High pass filters, DCT Transform (1D, 2D).

Unit 4

Image Restoration: Image Degradation/Restoration Process, Noise models, Noise Restoration Filters

Image Compression: Fundamentals of Image Compression, Huffman Coding, Run Length Coding, JPEG.

Unit 5

Morphological Image Processing: Erosion, Dilation, Opening, Closing, Hit-or-Miss Transformation, Basic Morphological Algorithms.

Unit 6

Image Segmentation: Point, Line and Edge Detection, Thresholding, Region Based Segmentation.

Practical

1. Write program to read and display digital image using MATLAB or SCILAB
 - a. Become familiar with SCILAB/MATLAB Basic commands
 - b. Read and display image in SCILAB/MATLAB
 - c. Resize given image
 - d. Convert given color image into gray-scale image
 - e. Convert given color/gray-scale image into black & white image
 - f. Draw image profile
 - g. Separate color image in three R G & B planes
 - h. Create color image using R, G and B three separate planes

- i. Flow control and LOOP in SCILAB
 - j. Write given 2-D data in image file
2. To write and execute image processing programs using point processing method
 - a. Obtain Negative image
 - b. Obtain Flip image
 - c. Thresholding
 - d. Contrast stretching
 3. To write and execute programs for image arithmetic operations
 - a. Addition of two images
 - b. Subtract one image from other image
 - c. Calculate mean value of image
 - d. Different Brightness by changing mean value
 4. To write and execute programs for image logical operations
 - a. AND operation between two images
 - b. OR operation between two images
 - c. Calculate intersection of two images
 - d. Water Marking using EX-OR operation
 - e. NOT operation (Negative image)
 5. To write a program for histogram calculation and equalization using
 - a. Standard MATLAB function
 - b. Program without using standard MATLAB functions
 - c. C Program
 6. To write and execute program for geometric transformation of image
 - a. Translation
 - b. Scaling
 - c. Rotation
 - d. Shrinking
 - e. Zooming
 7. To understand various image noise models and to write programs for
 - a. image restoration
 - b. Remove Salt and Pepper Noise
 - c. Minimize Gaussian noise
 - d. Median filter and Weiner filter

8. Write and execute programs to remove noise using spatial filters
 - a. Understand 1-D and 2-D convolution process
 - b. Use 3x3 Mask for low pass filter and high pass filter
9. Write and execute programs for image frequency domain filtering
 - a. Apply FFT on given image
 - b. Perform low pass and high pass filtering in frequency domain
 - c. Apply IFFT to reconstruct image
10. Write a program in C and MATLAB/SCILAB for edge detection using different edge detection mask
11. Write and execute program for image morphological operations erosion and dilation.
12. To write and execute program for wavelet transform on given image and perform inverse wavelet transform to reconstruct image.

References

1. Gonzalez, R. C., & Woods, R. E. (2017). *Digital Image Processing*. 4th edition. Pearson Education.
2. Jain, A. K. (1988). *Fundamentals of Digital Image Processing*. 1st edition Prentice Hall of India.

Additional Resources

1. Castleman, K. R. (1995.). *Digital Image Processing*. 1st edition. Pearson Education
2. Gonzalez, R. C., Woods, R. E., & Eddins, S. (2004). *Digital Image Processing using MATLAB*. Pearson Education Inc.
3. Schalkoff, D. (1989). *Image Processing and Computer Vision*. 1st edition. John Wiley and Sons.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Brightness, Adaptation and Discrimination, Light and Electromagnetic Spectrum, Image Sampling and Quantization.
2-5	Some Basic Relationships Between Pixels ,Spatial Domain Filtering, Intensity Transformation Functions, Histogram Equalization, Spatial Correlation and Convolution , Low pass filters, Order Statistics filters, Sharpening Spatial Filters: Laplacian filterFiltering in Frequency Domain The Discrete Fourier Transformation(DFT)
6-7	Frequency Domain Filtering:Ideal and Butterworth Low pass and High pass filters, Image Degradation/Restoration Process
8-10	Noise models, Noise Restoration Filters, Image Compression, Huffman Coding,Run Length Coding, Bit Plane Coding
11-12	Morphological Image Processing, Erosion, Dilation, Opening, Closing , Hit-or-Miss Transformation, Basic Morphological Algorithms
13-15	Image Segmentation: Point, Line and Edge Detection ,Thresholding, Region Based Segmentation

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

image transform, image restoration, image processing, image segmentation.

Microprocessors (BHCS16B) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

This course introduces internal architecture, programming model of Intel Microprocessors (8086 -Pentium) and assembly language programming using an assembler. Students will also learn interfacing of memory and I/O devices with microprocessor.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Describe the internal architecture of Intel microprocessors
2. Define and implement interfaces between the microprocessor and the devices.
3. Write assembly language programs

Detailed Syllabus

Unit 1

Microprocessor architecture: Internal architecture, Programming Model, Addressing modes, Data movement instructions

Unit 2

Microprocessor programming: Register Organization, instruction formats, Program control instructions, assembly language

Unit 3

Interfacing: Bus timings, Memory address decoding, cache memory and cache controllers, I/O interface, keyboard, timer, Interrupt controller, DMA controller, video controllers, communication interfaces.

Unit 4

Data transfer schemes: Synchronous data transfer, asynchronous data transfer, interrupt driven data transfer, DMA mode data transfer.

Unit 5

Microprocessor controllers: I/O controllers, interrupt controller, DMA controller, USART controller.

Unit 6

Advance microprocessor architecture: CISC architecture, RISC architecture, superscalar architecture, multicore architecture

Practical

ASSEMBLY LANGUAGE PROGRAMMING

1. Write a program for 32-bit binary division and multiplication
2. Write a program for 32-bit BCD addition and subtraction
3. Write a program for Linear search and binary search.

4. Write a program to add and subtract two arrays
5. Write a program for binary to ascii conversion
6. Write a program for ascii to binary conversion

References

1. Brey, B.B.(2009). *The Intel Microprocessors: Architecture, Programming and Interfacing*. 8th edition. Pearson Education.
2. Triebel, W.A., & Singh, A. (2002). *The 8088 and 8086 Microprocessors Programming, Interfacing, Software, Hardware and Applications*. 4th edition. Pearson Education.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-2	Microprocessor Architecture: Internal Architecture of microprocessor, Register Organization and flags, Programming models, Real mode memory addressing and protected mode memory addressing.
3-4	Addressing modes: Data memory addressing modes, program memory addressing modes, stack memory addressing mode.
5-6	Microprocessor Programming: Machine language, Instruction formats, Data movement instructions, assembly language syntax, Stack manipulation instructions, string transfer instructions, Arithmetic and logical instructions.
7-8	Program control instructions: The Jump group, different types of loops, defining function in assembly language, function call and return, introduction to interrupts.
9	Hardware Specification of 8086/8088: Pin-out diagrams of 8086/8088 microprocessors, function of pins, role of clock generator.
10	Memory Interfacing: Address decoding, interfacing of memory with 8088 and 8086.
11-12	I/O Interfacing: I/O port address decoding, isolated and memory mapped I/O, interfacing of keyboard and timer, communication interface

13-14	Interrupts : Purpose of interrupts, Interrupt instructions, interrupt vectors and interrupt descriptors, functioning of interrupt controller
15	Direct Memory Access (DMA): Basic DMA operation, functioning of DMA controller

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Microprocessor architecture, microprocessor programming, interfacing,

Information Security (BHCS17A) Discipline Specific Course - (DSE)

Credit: 06

Course Objective

The course offers a broad overview of the fundamentals of information security covering topics such as error correction/detection, cryptography, steganography, malwares, This course also touches on the implications of security in Internet of Things (IoT).

Course Learning Outcomes

On successful completion of this course, a student will be able to,

1. Identify the major types of threats to information security
2. Describe the role of cryptography in security
3. Select appropriate error-detection and error-correction methods for an application
4. Discuss the strengths and weaknesses of private and public key crypto systems
5. Describe malwares and memory exploits
6. Discuss the need for security in IoT

Detailed Syllabus

Unit 1

Introduction: Security Concepts, Challenges, Security architecture, Security attacks, security services, security mechanisms

Unit 2

Error detecting/correction: Block Codes, Generator Matrix, Parity Check Matrix, Minimum distance of a Code, Error detection and correction, Standard Array and syndrome decoding, Hamming Codes

Unit 3

Cryptography: Encryption, Decryption, Substitution and Transposition, Confusion and diffusion, Symmetric and Asymmetric encryption, Stream and Block ciphers, DES, cryptanalysis.

Public-key cryptography, Diffie-Hellman key exchange, man-in-the-middle attack

Digital signature, Steganography, Watermarking.

Unit 4

Malicious software's: Types of malwares (viruses, worms, trojan horse, rootkits, bots), Memory exploits - Buffer overflow, Integer overflow

Unit 5

Security in Internet-of-Things: Security implications, Mobile device security - threats and strategies

Practical

1. Implement the error correcting code.
2. Implement the error detecting code.
3. Implement caesar cipher substitution operation.
4. Implement monoalphabetic and polyalphabetic cipher substitution operation.
5. Implement playfair cipher substitution operation.
6. Implement hill cipher substitution operation.
7. Implement rail fence cipher transposition operation.
8. Implement row transposition cipher transposition operation.
9. Implement product cipher transposition operation.
10. Illustrate the Ciphertext only and Known plaintext attacks.
11. Implement a stream cipher technique

References

1. Pfleeger, C.P., Pfleeger,S.L., & Margulies, J. (2015). *Security in Computing*. 5th edition. Prentice Hall
2. Lin, S. & Costello, D. J. (2004). *Error Control Coding: Fundamentals and applications*. 2nd edition. Pearson Education
3. Stallings, W. (2018). *Cryptography and network security*. 7th edition. Pearson Education.

Additional Resources

1. Berlekamp, E. R. (1986). *Algebraic Coding Theory*. McGraw Hill Book Company
2. Stallings, W. (2018) *Network security, essentials*. 6th edition. Pearson Education.
3. Whitman M.E., & Mattord H.J. (2017). *Principle of Information Security*. 6th edition. Cengage Learning.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-2	Security Concepts, Challenges, Security architecture, Security attacks, security services, security mechanisms
3-4	Error detecting/correction, Block Codes, Generator Matrix, Parity Check Matrix, Minimum distance of a Code, Error detection and correction, Standard Array and syndrome decoding, Hamming Codes
5-7	Cryptography: Encryption, Decryption, Substitution and Transposition, Confusion and diffusion, Symmetric and Asymmetric encryption, Stream and Block ciphers, DES, Modes of DES
8-9	Cryptanalysis, Types of cryptanalytic attacks, Public-key cryptography, Diffie-Hellman key exchange, man-in-the-middle attack

10-11	Digital signatures, Steganography and Digital Watermarking
12-13	Malicious Software: Types of malwares (viruses, worms, trojan horse, rootkits, bots), Memory exploits - Buffer overflow, Integer overflow
14-15	Security in Internet-of-Things, Security implications, Mobile device security - threats and strategies, Cyberlaws

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Security mechanisms, private and public key cryptography, malware detection, security in IoT.

Data Mining (BHCS17B) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

This course introduces data mining techniques and enables students to apply these techniques on real-life datasets. The course focuses on three main data mining techniques: Classification, Clustering and Association Rule Mining tasks.

Course Learning Outcomes

On successful completion of the course, students will be able to do following:

1. Pre-process the data, and perform cleaning and transformation.
2. Apply suitable classification algorithm to train the classifier and evaluate its performance.
3. Apply appropriate clustering algorithm to cluster data and evaluate clustering quality
4. Use association rule mining algorithms and generate frequent item-sets and association rules

Detailed Syllabus

Unit 1

Introduction to Data Mining - Applications of data mining, data mining tasks, motivation and challenges, types of data attributes and measurements, data quality.

Data Pre-processing - aggregation, sampling, dimensionality reduction, Feature Subset Selection, Feature Creation, Discretization and Binarization, Variable Transformation.

Unit 2

Classification: Basic Concepts, Decision Tree Classifier: Decision tree algorithm, attribute selection measures, Nearest Neighbour Classifier, Bayes Theorem and Naive Bayes Classifier,

Model Evaluation: Holdout Method, Random Sub Sampling, Cross-Validation, evaluation metrics, confusion matrix.

Unit 3

Association rule mining: Transaction data-set, Frequent Itemset, Support measure, Apriori Principle, Apriori Algorithm, Computational Complexity, Rule Generation, Confidence of association rule.

Unit 4

Cluster Analysis: Basic Concepts, Different Types of Clustering Methods, Different Types of Clusters, K-means: The Basic K-means Algorithm, Strengths and Weaknesses of K-means algorithm, Agglomerative Hierarchical Clustering: Basic Algorithm, Proximity between clusters, DBSCAN: The DBSCAN Algorithm, Strengths and Weaknesses.

Practical

Section 1: Preprocessing

Q1. Create a file “people.txt” with the following data:

Age	agegroup	height	status	yearsmarried
21	adult	6.0	single	-1
2	child	3	married	0
18	adult	5.7	married	20
221	elderly	5	widowed	2
34	child	-7	married	3

i) Read the data from the file “people.txt”.

ii) Create a ruleset E that contain rules to check for the following conditions:

1. The age should be in the range 0-150.
2. The age should be greater than yearsmarried.

3. The status should be married or single or widowed.
4. If age is less than 18 the agegroup should be child, if age is between 18 and 65 the agegroup should be adult, if age is more than 65 the agegroup should be elderly.

iii) Check whether ruleset E is violated by the data in the file `people.txt`.

iv) Summarize the results obtained in part (iii)

v) Visualize the results obtained in part (iii)

Q2. Perform the following preprocessing tasks on the `dirty_iris` datasetⁱⁱ.

1. Calculate the number and percentage of observations that are complete.
2. Replace all the special values in data with NA.
3. Define these rules in a separate text file and read them.
(Use `editfile` function in R (package `editrules`). Use similar function in Python).
Print the resulting constraint object.
 - Species should be one of the following values: `setosa`, `versicolor` or `virginica`.
 - All measured numerical properties of an iris should be positive.
 - The petal length of an iris is at least 2 times its petal width.
 - The sepal length of an iris cannot exceed 30 cm.
 - The sepals of an iris are longer than its petals.
4. Determine how often each rule is broken (`violatedEdits`). Also summarize and plot the result.

Find outliers in sepal length using `boxplot` and `boxplot.stats`

Q3. Load the data from wine dataset. Check whether all attributes are standardized or not (mean is 0 and standard deviation is 1). If not, standardize the attributes. Do the same with Iris dataset.

Section 2: Data Mining Techniques

Run following algorithms on 2 real datasets and use appropriate evaluation measures to compute correctness of obtained patterns:

Q4. Run Apriori algorithm to find frequent itemsets and association rules

4.1 Use minimum support as 50% and minimum confidence as 75%

4.2 Use minimum support as 60% and minimum confidence as 60 %

Q5. Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set into training and test set. Compare the accuracy of the different classifiers under the following situations:

5.1 a) Training set = 75% Test set = 25%

b) Training set = 66.6% (2/3rd of total), Test set = 33.3%

5.2 Training set is chosen by i) hold out method ii) Random subsampling iii) Cross-Validation. Compare the accuracy of the classifiers obtained.

5.3 Data is scaled to standard format.

Q6. Use Simple Kmeans, DBScan, Hierarchical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.

Recommended Datasets for DataMining practicals

1. UCI Machine Learning repository.
2. KDD Datasets
3. Open data platform, Government of India (<https://data.gov.in/>)

References

1. Han, J., Kamber, M., & Jian, P. (2011). *Data Mining: Concepts and Techniques*. 3rd edition. Morgan Kaufmann
2. Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining*. 1st Edition. Pearson Education.

Additional Resources

1. Gupta, G. K. (2006). *Introduction to Data Mining with Case Studies*. Prentice-Hall of India.
2. Hand, D., & Mannila, H. & Smyth, P. (2006). *Principles of Data Mining*. Prentice-Hall of India.
3. Pujari, A. (2008). *Data Mining Techniques*. 2nd edition. Universities Press.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class-room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Introduction to Data Mining , Challenges , Data Mining Origins, Data Mining Tasks, Applications
2-3	Types of data, Data Quality, Data Pre-processing, Measures of similarity and dissimilarity
5-8	Classification - Preliminaries, General Approach to Solving a Classification Problem, Decision Tree Induction , Evaluating the Performance of a Classifier
8-9	Rule Based Classifier , Nearest Neighbor Classifiers, Bayesian Classifiers
10-11	Association Rules -Problem definition, Frequent item-set generation (Apriori algorithm), Rule generation
11-12	Clustering - Basic concepts of clustering analysis, K-Means
13-14	Agglomerative Hierarchical Clustering, DBSCAN
15	Quality of clustering

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

data mining, classifiers, data pre-processing, metrics.

Advanced Algorithms (BHCS17C) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

This course focuses on the study of advanced data structures and algorithms for solving problems efficiently and their theoretical behavior. The course also includes study of network flow algorithms, NP completeness and backtracking.

Course Learning Outcomes

On successful completion of this course, the student will be able to:

1. Implement and empirically analyze advanced data-structures like tries, suffix trees.
2. Apply amortized analysis.
3. Develop more sophisticated algorithms using techniques like divide and conquer, dynamic programming, greedy strategy, and augmentation
4. Prove that certain problems are too hard to admit fast solutions.
5. Develop algorithms using backtracking for the hard problems.

Detailed Syllabus

Unit 1

Advanced Data Structures: Skip Lists, Red-Black trees, Splay Trees, Mergeable heaps (Fibonacci heaps), DS for sets - Union-Find Data Structure, Dynamic Tables, Dictionaries, Data structures for strings - Tries, Suffix trees.

Unit 2

Divide and Conquer: Counting Inversions, Closest pair of points, Integer Multiplication,

Unit 3

Greedy Algorithm: Interval Scheduling, Huffman Code, Correctness and Analysis,

Unit 4

Dynamic Programming: Segmented Least Squares, Shortest Paths, Negative Cycles in Graphs

Unit 5

Network Flows: Max-flow problem, Ford Fulkerson Algorithm, Maximum flows and Minimum Cuts in a network, Bipartite Matching.

Unit 6

NP Completeness: Polynomial time reductions, Efficient Certification and Definition of NP, NP Complete problems, Sequencing problems, Partitioning problems, co-NP and asymmetry of NP.

Backtracking: Constructing All Subsets, Constructing All Permutations, Constructing All Paths in a Graph.

Practical

Tutorials based on Theory.

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., & Stein, C. (2010). *Introduction to Algorithms*. 3rd edition. Prentice-Hall of India Learning Pvt. Ltd.
2. Kleinberg, J., & Tardos, E. (2013). *Algorithm Design*. 1st edition. Pearson Education India.

Additional Resources

1. Basse, S., & Gleder, A. V. (1999). *Computer Algorithm – Introduction to Design and Analysis*. 3rd edition. Pearson Education.
2. Dasgupta, S., Papadimitriou, C., & Vazirani, U. (2017). *Algorithms*. 1st edition. TataMcGraw Hill.
3. Skiena, S. S. (2008). *The Algorithm Design Manual*. 2nd edition. Springer-Verlag London

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-4	Advanced Data Structures: Skip Lists, Red-Black trees, Splay Trees, Mergeable heaps (Fibonacci heaps), DS for sets - Union-Find Data Structure, Dynamic Tables, Dictionaries, Data structures for strings - Tries, Suffix trees
5	Divide and Conquer: Counting Inversions, Closest pair of points, Integer Multiplication
6-7	Greedy Algorithm: Interval Scheduling, Huffman Code, Correctness and Analysis
8-9	Dynamic Programming: Segmented Least Squares, Shortest Paths, Negative Cycles in Graphs
10-11	Network Flows: Max-flow problem, Ford Fulkerson Algorithm, Maximum flows and Minimum Cuts in a network, Bipartite Matching

12-13	NP Completeness: Polynomial time reductions, Efficient Certification and Definition of NP, NP Complete problems, Sequencing problems, Partitioning problems, co-NP and asymmetry of NP
14-15	Backtracking: Constructing All Subsets, Constructing All Permutations, Constructing All Paths in a Graph

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Algorithms, Analysis, Network Flows, NP Completeness.

Machine Learning (BHCS18A) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

The course aims at introducing the basic concepts and techniques of machine learning so that a student can apply machine learning techniques to a problem at hand.

Course Learning Outcomes

On successful completion of this course, the student will be able to:

1. Differentiate between supervised and unsupervised learning tasks.
2. Differentiate between linear and non-linear classifiers.
3. Describe theoretical basis of SVM
4. Implement various machine learning algorithms learnt in the course.

Detailed Syllabus

Unit 1

Introduction: Basic definitions, Hypothesis space and inductive bias, Bayes optimal classifier and Bayes error, Occam's razor, Curse of dimensionality, dimensionality reduction, feature scaling, feature selection methods.

Unit 2

Regression: Linear regression with one variable, linear regression with multiple variables, gradient descent, logistic regression, over-fitting, regularization. performance evaluation metrics, validation methods.

Unit 3

Classification: Decision trees, Naive Bayes classifier, k-nearest neighbor classifier, perceptron, multilayer perceptron, neural networks, back-propagation algorithm, Support Vector Machine (SVM), Kernel functions.

Unit 4

Clustering: Approaches for clustering, distance metrics, K-means clustering, expectation maximization, hierarchical clustering, performance evaluation metrics, validation methods.

Practical

For practical Labs for Machine Learning, students may use softwares like MABLAB/Octave or Python. For later exercises, students can create/use their own datasets or utilize datasets from online

repositories like UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/>).

1. Perform elementary mathematical operations in Octave/MATLAB like addition, multiplication, division and exponentiation.
2. Perform elementary logical operations in Octave/MATLAB (like OR, AND, Checking for Equality, NOT, XOR).
3. Create, initialize and display simple variables and simple strings and use simple formatting for variable.
4. Create/Define single dimension / multi-dimension arrays, and arrays with specific values like array of all ones, all zeros, array with random values within a range, or a diagonal matrix.
5. Use command to compute the size of a matrix, size/length of a particular row/column, load data from a text file, store matrix data to a text file, finding out variables and their features in the current scope.

6. Perform basic operations on matrices (like addition, subtraction, multiplication) and display specific rows or columns of the matrix.
7. Perform other matrix operations like converting matrix data to absolute values, taking the negative of matrix values, adding/removing rows/columns from a matrix, finding the maximum or minimum values in a matrix or in a row/column, and finding the sum of some/all elements in a matrix.
8. Create various type of plots/charts like histograms, plot based on sine/cosine function based on data from a matrix. Further label different axes in a plot and data in a plot.
9. Generate different subplots from a given plot and color plot data.
10. Use conditional statements and different type of loops based on simple example/s.
11. Perform vectorized implementation of simple matrix operation like finding the transpose of a matrix, adding, subtracting or multiplying two matrices.
12. Implement Linear Regression problem. For example, based on a dataset comprising of existing set of prices and area/size of the houses, predict the estimated price of a given house.
13. Based on multiple features/variables perform Linear Regression. For example, based on a number of additional features like number of bedrooms, servant room, number of balconies, number of houses of years a house has been built – predict the price of a house.
14. Implement a classification/ logistic regression problem. For example based on different features of students data, classify, whether a student is suitable for a particular activity. Based on the available dataset, a student can also implement another classification problem like checking whether an email is spam or not.
15. Use some function for regularization of dataset based on problem 14.
16. Use some function for neural networks, like Stochastic Gradient Descent or backpropagation - algorithm to predict the value of a variable based on the dataset of problem 14.

References

1. Flach, P. (2015). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press.
2. Mitchell, T.M. (2017). *Machine Learning*. McGraw Hill Education.

Additional References:

1. Christopher & Bishop, M. (2016). *Pattern Recognition and Machine Learning*. New York: Springer-Verlag

2. Haykins, S.O. (2010). *Neural Networks and Learning Machines*. 3rd edition. PHI.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Basic definitions, Hypothesis space and inductive bias, Bayes optimal classifier and Bayes error, Occam's razor
2	Curse of dimensionality, dimensionality reduction, feature scaling, feature selection methods
3	Linear regression with one variable, linear regression with multiple variables
4 -5	Gradient descent, logistic regression, over-fitting, regularization
6	Performance evaluation metrics, validation methods.
7	Decision trees, Naive Bayes classifier ,k-nearest neighbor classifier
8 - 9	Perceptron, Multilayer perceptron, neural networks, back-propagation algorithm
10-11	Support Vector Machine (SVM), Kernel functions
12	Approaches for clustering, distance metrics
13	K-means clustering, expectation maximization
14	Hierarchical clustering
15	Clustering validation methods, performance evaluation metrics

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Machine learning, unsupervised learning, supervised learning, support vector machines, neural networks, classification, clustering,

Deep Learning (BHCS18B) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

The objective of this course is to introduce students to deep learning algorithms and their applications in order to solve real problems.

Course Learning Outcomes

On successful completion of this course, the student will be able to:

1. Describe the feed-forward and deep networks.
2. Design single and multi-layer feed-forward deep networks and tune various hyper-parameters.
3. Implement deep neural networks to solve a problem
4. Analyse performance of deep networks.

Detailed Syllabus

Unit 1

Introduction: Historical context and motivation for deep learning; basic supervised classification task, optimizing logistic classifier using gradient descent, stochastic gradient descent, momentum, and adaptive sub-gradient method.

Unit 2

Neural Networks: Feedforward neural networks, deep networks, regularizing a deep network, model exploration, and hyper parameter tuning.

Unit 3

Convolution Neural Networks: Introduction to convolution neural networks: stacking, striding and pooling, applications like image, and text classification.

Unit 4

Sequence Modeling: Recurrent Nets: Unfolding computational graphs, recurrent neural networks (RNNs), bidirectional RNNs, encoder-decoder sequence to sequence architectures, deep recurrent networks, LSTM networks.

Unit 5

Autoencoders: Undercomplete autoencoders, regularized autoencoders, sparse autoencoders, denoising autoencoders, representational power, layer, size, and depth of autoencoders, stochastic encoders and decoders.

Unit 6

Structuring Machine Learning Projects: Orthogonalization, evaluation metrics, train/dev/test distributions, size of the dev and test sets, cleaning up incorrectly labeled data, bias and variance with mismatched data distributions, transfer learning, multi-task learning.

Practical

1. Implement logistic regression classification with (a) gradient descent and (b) stochastic gradient descent method. Plot cost function over iteration.
2. Experiment with logistic regression by adding momentum term, and adaptive sub-gradient method
3. Write the code to learn weights of a perceptron for Boolean functions (NOT, OR, AND, NOR, and NAND).
4. Implement a feed-forward neural network for solving (a) regression and (b) 2-class classification problem. Also experiment with hyper-parameter tuning.
5. Train and test a feed-forward neural network for multi-class classification using softmax layer as output.
6. Create a 2D and 3D CNN for image classification. Experiment with different depth of network, striding and pooling values.
7. Implement (a) RNN for image classification, (b) GRU network and (c) Implement LSTM networks
8. Implement an auto-encoder, denoising autoencoders and sparse autoencoders.
9. Design a stochastic encoders and decoders.

References:

1. Bunduma, N. (2017). *Fundamentals of Deep Learning*. O'reilly Books.
2. Heaton, J.(2015). *Deep Learning and Neural Networks*, Heaton Research Inc.

Additional References:

1. Goodfellow, I. (2016). *Deep Learning*. MIT Press.
2. Deng, L., & Yu, D. (2009). *Deep Learning: Methods and Applications (Foundations and Trends in Signal Processing)*. Publishers Inc.
3. Hall, M.L, (2011). *Deep Learning*. VDM Verlag

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	Introduction: Historical context and motivation for deep learning; basic supervised classification task
2-3	Optimizing logistic classifier using gradient descent, stochastic gradient descent, momentum, and adaptive sub-gradient method
4-5	Neural Networks: Feedforward neural networks, deep networks, regularizing a deep network, model exploration, and hyper parameter tuning
6-7	Convolution Neural Networks: Introduction to convolution neural networks: stacking, striding and pooling, applications like image, and text classification
8	Sequence Modeling: Recurrent Nets: Unfolding computational graphs, recurrent neural networks (RNNs), bidirectional RNNs
9	Encoder-decoder sequence to sequence architectures, deep recurrent networks, LSTM networks
10	Autoencoders: Undercomplete autoencoders, regularized autoencoders, sparse autoencoders
11-12	Denoising autoencoders, representational power, layer, size, and depth of autoencoders, stochastic encoders and decoders.
13	Structuring Machine Learning Projects: Orthogonalization, evaluation metrics, train/dev/test distributions,
14-15	Size of the dev and test sets, cleaning up incorrectly labeled data, bias and variance with mismatched data distributions, transfer learning, multi-task learning.

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Convolution Neural Networks, Recurrent nets, autoencoders

Unix Network Programming (BHCS18C) Discipline Specific Elective - (DSE)

Credit: 06

Course Objective

This course introduces the concepts of Internet protocols, ports used during communication, Client/Server concepts and various transport protocols used in computer network applications and services. The objective is to equip the students with technical knowledge of it comprises of the study of the sockets used with TCP and UDP include IPV4 & IPV6.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Describe and analyse the various Internet Transport layer protocols used in TCP/IP AND UDP.
2. Comprehend the concepts and structures of both TCP based connection-oriented and UDP based connection-less client server applications.
3. Write various real-life client-server applications using socket programming.
4. Modify, maintain and extend the present internet client-server applications and write any new type of internet applications to suit the current needs of Internet users.

Detailed Syllabus

Unit 1

Introduction: Basics of Client Server applications, Example of day time client server, concurrent servers, protocols, sockets, port numbers.

Unit 2

Connection-oriented and Connection-less client server Applications: Elementary TCP sockets – Socket, connect, bind, listen, accept, fork and exec function, close function, Socket Address Structures, Byte Ordering and Manipulation Functions, TCP Client and Server for Echo, Signal Handling in case of crashing and rebooting of server, Shutdown process function

Unit 3

Socket Options: Getsockopt and stockpot functions, Socket states, Generic socket option

Unit 4

Connection-oriented and connection-less Sockets: TCP-oriented basic concurrent client server applications, UDP oriented Echo client and server application, Handling of errors like lost datagram, Lack of flow control with UDP, determining outgoing interface with UDP.

Unit 5

Elementary name and Address conversions: Domain Name System, socket functions like gethostbyname, gethostbyname2, gethostbyaddr function, uname function, gethostname function, getservbyname and getservbyport functions.

Unit 6

Advanced Sockets: Daemon Processes, Multithreaded server, Raw sockets.

Practical

1. Implement TCP Echo client and TCP Echo server (Iterative).
2. Implement TCP Echo client and TCP Echo server (Concurrent).
3. Implement TCP daytime client and TCP daytime server (Iterative).
4. Implement TCP daytime client and TCP daytime server (concurrent).
5. Implement UDP Echo Client and UDP Echo Server.
6. Implement UDP daytime Client and UDP daytime server.
7. Implement TCP client and server (concurrent) where client gets input from the user and sends it to server. Server displays it on the screen. Server then gets another input from the user and sends it to client. Client displays it on the screen. The process continues till server or client sends “bye” to the other party.
8. Implement TCP client and server (concurrent) where client requests server to transfer a file. Assume file is smaller than 1K size. If the file is present on the server, it is sent to the client otherwise an error message is sent to client. Client copies the file on the hard disk and disconnects.

9. Implement UDP client and UDP server where server displays the IP address and port number of the client sending the datagram. Client sends a datagram (size 64 bytes) three times to the same server. Server sends the message back to client. Client reports the time elapsed in sending and receiving of the message. Use connected UDP sockets.
10. Write to program to
 1. display name of the host
 2. all IP addresses of the host.
 3. Check whether FTP and HTTP services are running on the system.
 4. Display the name of the service running on port number specified by user.

References

1. Stevens, R. W., Fenner, B., & Rudoff, A. M. (2010). *Unix Network Programming: The Sockets Networking API*. 3rd edition. PHI.

Additional Resources:

1. Forouzan, B. A. (2017). *Data Communication and Networking*. 4th edition. McGraw-Hill Education.
2. Stevens, R. W. (2009). *Unix Network Programming*. 1st edition. PHI.
3. Tanenbaum, A. S. (2012). *Computer Networks*. 5th edition. Pearson Education

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class-room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Contents
1	Introduction, client server applications, protocols, port numbers
2-3	Sockets Functions, fork and exec function, Socket address structure
4	TCP Echo Server

5	Signal Handling
6-7	I/O Multiplexing
8-9	Socket Options, Getsockopt and stockpot functions, socket states, generic socket options
10	Elementary UDP sockets, TCP and UDP oriented client server applications
11	Elementary name and Address conversions, DNS, socket functions
12	Daemon Processes
13-14	Multithreaded server
15	Raw sockets

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Connection oriented sockets, connection less sockets, advanced sockets

Project Work / Dissertation (BHCS18D) Discipline Specific Elective - (DSE) Credit: 06

Course Objective

The students will undergo one semester of project work based on the concepts studied in a subject of their choice. The objective is to train the students for the industry by exposing them to prototype development of real life software.

Course Learning Outcomes

On successful completion of this course, a student will be able to:

1. develop a project plan based on informal description of the project.
2. implement the project as a team.
3. write a report on the project work carried out by the team and defend the work done by the team collectively.
4. present the work done by the team to the evaluation committee.

Unit 1

The students will work on any project based on the concepts studied in core/elective/ skill based elective courses. Specifically, the project could be a research study, or a software development project.

Unit 2

Project Group Organization/Plan

- Students will initially prepare a synopsis (500 words) and submit it to their respective department.
- For a given project, the group size could be a maximum of four (04) students.
- Each group will be assigned a teacher as a supervisor who will be responsible for their lab classes.
- A maximum of four (04) projects would be assigned to one teacher.

Unit 3

Project Evaluation

- 100 marks for end semester examination comprising Viva/presentation (50 marks) and project report evaluation (50 marks): to be awarded jointly by the examiner and supervisor / mentor.
- 50 marks for continuous evaluation (to be awarded by the supervisor/mentor). Work carried out in each lab session will be assessed out of five marks (zero for being absent). Finally, the marks obtained will be scaled out of a maximum of 50 marks. For example, if

30 lab sessions are held in a semester, and a student has obtained an aggregate of 110 marks, then he/she will be assigned round $(110/(30*5))$ i.e. 37 marks.

- The students will submit only the soft copies of the report.
- The reports may be retained by the examiners.

Practical

Practical/discussion sessions based on the area of the project.

Teaching Learning Process

- Group Discussions
- Presentations by group of students for enhanced learning.

Assessment Methods

- Assignments, presentations, viva, quiz
- Internal assessment
- End semester exam

Keywords

Software Development, Project planning.

Web Design and development (BHCS19A) Skill-Enhancement Elective Course - (SEC)

Credit: 04

Course Objective

This course will introduce students to the fundamental concepts of web development. This course will equip students with the ability to design and develop a dynamic website using technologies like HTML, CSS, JavaScript, PHP and MySQL on platform like WAMP/XAMP/LAMP.

Course Learning Outcomes

On successful completion of the course, students will be able:

1. Design and develop a website
2. Use Front end technologies like HTML, CSS and JavaScript
3. Use backend technologies like PHP and MySQL
4. Work on platforms like WAMP/XAMP/LAMP

Detailed Syllabus

Unit 1

Introduction to Static and Dynamic Websites (Website Designing and Anatomy of Webpage)

Unit 2

Introduction to HTML and CSS (Basic Tags, Lists, Handling Graphics, Tables, Linking, Frames, Forms), Introduction to DOM

Unit 3

Introduction to JavaScript (Basic Programming Techniques & Constructs, GET/POST Methods, Operators, Functions, DOM Event handling, Forms Validation, Cookies), Inter-page communication and form data handling using JavaScript

Unit 4

Introduction to PHP (Working, Difference with other technologies like JSP and ASP), PHP Programming Techniques (Data types, Operators, Arrays, Loops, Conditional statements, Functions, Regular expressions)

Unit 5

Form Data Handling with PHP, Database connectivity and handling using PHP-MySQL

Practical

1. Practicals based on HTML
2. Practicals based on CSS
3. Practicals based on PHP
4. Practicals to create HTML forms
5. Practicals based on database connectivity with

References:

1. Bayross, I. (2013). *Web enabled commercial application development using HTML, JavaScript, DHTML and PHP*. 4th edition. BPB Publication.

2. Holzner, S.(2007). *PHP: The Complete Reference Paperback*, McGraw Hill Education (India).

Additional Resources

1. Boronczyk, T., & Psinas, M. E. (2008). *PHP and MYSQL (Create-Modify-Reuse)*. Wiley India Private Limited.
2. Welling, L., & Thompson, L. (2008). *PHP and MySQL Web Development*. 4th edition. Addition Paperback, Addison-Wesley Professional.
3. Nixon, R. (2014). *Learning PHP, MySQL, JavaScript, CSS & HTML5*. 3rd edition. Paperback, O'reilly Media
4. Sklar, D., & Trachtenberg, A., (2014). *PHP Cookbook: Solutions & Examples for PHP Programmers*. 2nd edition. O'reilly Media

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class-room teaching methods
- Interactive sessions
- Class discussions

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Static and dynamic websites, form handling, database connectivity.

Programming in Python (BHCS19B) Skill-Enhancement Elective Course - (SEC)

Credit: 06

Course Objective

This course is designed to introduce the student to the basics of programming using Python. The course covers the topics essential for developing well documented modular programs using different instructions and built-in data structures available in Python.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Develop, document, and debug modular python programs to solve computational problems.
2. Select a suitable programming construct and data structure for a situation.
3. Use built-in strings, lists, sets, tuples and dictionary in applications.
4. Define classes and use them in applications.
5. Use files for I/O operations.

Detailed Syllabus

Unit 1

Introduction to Programming using Python: Structure of a Python Program, Functions, Interpreter shell, Indentation. Identifiers and keywords, Literals, Strings, Basic operators (Arithmetic operator, Relational operator, Logical or Boolean operator, Assignment Operator, Bit wise operator).

Unit 2

Building blocks of Python: Standard libraries in Python, notion of class, object and method.

Unit 3

Creating Python Programs: Input and Output Statements, Control statements:-branching, looping, Exit function, break, continue and pass, mutable and immutable structures. Testing and debugging a program

Unit 4

Built-in data structures: Strings, lists, Sets, Tuples and Dictionary and associated operations. Basic searching and sorting methods using iteration and recursion.

Unit 5

Visualization using 2D and 3D graphics: Visualization using graphical objects like Point, Line, Histogram, Sine and Cosine Curve, 3D objects

Unit 6

Exception Handling and File Handling: Reading and writing text and structured files, Errors and Exceptions.

Practical

1. Execution of expressions involving arithmetic, relational, logical, and bitwise operators

in the shell window of Python IDLE.

2. Write a Python function to produce the outputs such as:

a) *

 *

(b) 1
 232
 34543
 4567654
 567898765

3. Write a Python program to illustrate the various functions of the “Math” module.

4. Write a function that takes the lengths of three sides: **side1**, **side2** and **side3** of the triangle as the input from the user using **input** function and return the area of the triangle as the output. Also, assert that sum of the length of any two sides is greater than the third side.
5. Consider a showroom of electronic products, where there are various salesmen. Each salesman is given a commission of 5%, depending on the sales made per month. In case the sale done is less than 50000, then the salesman is not given any commission. Write a function to calculate total sales of a salesman in a month, commission and remarks for the salesman. Sales done by each salesman per week is to be provided as input. Assign remarks according to the following criteria:
 - Excellent: Sales ≥ 80000
 - Good: Sales ≥ 60000 and < 80000
 - Average: Sales ≥ 40000 and < 60000
 - Work Hard: Sales < 40000
6. Write a Python function that takes a number as an input from the user and computes its factorial.
7. Write a Python function to return nth terms of Fibonacci sequence
8. Write a function that takes a number with two or more digits as an input and finds its reverse and computes the sum of its digits.
9. Write a function that takes two numbers as input parameters and returns their least common multiple and highest common factor.
10. Write a function that takes a number as an input and determine whether it is prime or not.
11. Write a function that finds the sum of the n terms of the following series:
 - a) $1 - x^2/2! + x^4/4! - x^6/6! + \dots x^n/n!$

- b) $1 + x^2/2! + x^4/4! + x^6/6! + \dots x^n/n!$
12. Write a Python function that takes two strings as an input from the user and counts the number of matching characters in the given pair of strings.
 13. Write a Python function that takes a string as an input from the user and displays its reverse.
 14. Write a Python function that takes a string as an input from the user and determines whether it is palindrome or not.
 15. Write a Python function to calculate the sum and product of two compatible matrices
 16. Write a function that takes a list of numbers as input from the user and produces the corresponding cumulative list where each element in the list present at index i is the sum of elements at index $j \leq i$.
 17. Write a function that takes n as an input and creates a list of n lists such that i th list contains first five multiples of i .
 18. Write a function that takes a sentence as input from the user and calculates the frequency of each letter. Use a variable of dictionary type to maintain the count.
 19. Write a Python function that takes a dictionary of *word:meaning* pairs as an input from the user and creates an inverted dictionary of the form meaning:list-of-words.
 20. Usage of Python debugger tool-pydb and PythonTutor.
 21. Implementation of Linear and binary search techniques
 22. Implementation of selection sort, insertion sort, and bubble sort techniques
 23. Write a menu-driven program to create mathematical 3D objects
Curve, Sphere, Cone, Arrow, Ring, Cylinder.
 24. Write a program that makes use of a function to accept a list of n integers and displays a histogram.
 25. Write a program that makes use of a function to display sine, cosine, polynomial and exponential curves.
 26. Write a program that makes use of a function to plot a graph of people with pulse rate p vs. height h . The values of p and h are to be entered by the user.
 27. Write a function that reads a file **file1** and displays the number of words and the number of vowels in the file.
 28. Write a Python function that copies the content of one file to another.
 29. Write a function that reads a file **file1** and copies only alternative lines to another file **file2**. Alternative lines copied should be the odd numbered lines.

References

1. Downey, A.B., (2015), *Think Python—How to think like a Computer Scientist*, 3rd edition. O'Reilly Media.
2. Taneja, S. & Kumar, N., (2017), *Python Programming- A Modular Approach*. Pearson Education.

Additional Resources

1. Brown, M. C. (2001). *The Complete Reference: Python*, McGraw Hill Education.
2. Dromey, R. G. (2006), *How to Solve it by Computer*, Pearson Education.
3. Guttag, J.V.(2016), *Introduction to computation and programming using Python*. MIT Press.
4. Liang, Y.D. (2013), *Introduction to programming using Python*. Pearson Education.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class-room teaching methods
- Interactive sessions
- Class discussions
- Mini projects in the laboratory

Tentative weekly teaching plan is as follows:

Week	Content
1.	Python Programming: An Introduction Structure of a Python program, understanding Python interpreter/Python shell, indentation. Atoms, identifiers and keywords, literals, Python strings, arithmetic operator, relational operator, logical or boolean operator, bit wise operators.
2	Variables and Functions Python standard libraries such as sys and math. Variables and assignment statements. Built-in functions such as input and print.
3-4	Control Structures if conditional statement and for loop, While loop, break, continue, and pass statement, else statement. Infinite loop
5	Functions Function definition and call, default parameter values, keyword arguments, assert statement
6	Strings and Lists Strings-slicing, membership, and built-in functions on strings Lists- list operations.

7.	Mutable object Lists- built-in functions, list comprehension, passing list as arguments, copying list objects.
8	Sets, tuples, and dictionary- associated operations and built-in functions.
9	Testing and Debugging Determining test cases, use of python debugger tool- pydb for debugging
10	Searching and Sorting Linear search, binary search, selection sort, insertion sort, and bubble sort
11	Python 2D and 3D Graphics Visualization using graphical objects like point, line, histogram, sine and cosine curve, 3D objects
12	File Handling Reading and writing text and structured files.
13	Errors and Exceptions Types of errors and exceptions, and exception handling
14	Classes Notion of class, object, and method.

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Python Program, Control structure, Decision making, Functions, Strings, Lists, Dictionary.

Android Programming (BHCS20A) Skill-Enhancement Elective Course - (SEC)

Credit:04

Course Objective

The paper provides an introduction to development of mobile application on android platform. The topics include the Android development environment, activities, fragments, user interfaces, intents, broadcast sender/receivers, services, notifications, SQLite database handling.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Describe characteristics of Android operating system
2. Describe components of an android applications
3. Design user interfaces using various widgets, dialog boxes, menus
4. Define interaction among various activities/applications using intents, broadcasting, services
5. Develop Android applications that require database handling

Detailed Syllabus

Unit 1

Introduction: Review to JAVA & OOPS Concepts, History of Android, Introduction to Android Operating Systems, Android Development Tools, Android Architecture, Android components including activities, view and view group, services, content providers, broadcast receivers, intents, parcels, instance state.

Unit 2

User Interface Architecture: application context, intents: explicit intents, returning results from activities, implicit intents, intent filter and intent resolution, and applications of implicit intents, activity life cycle, activity stack, application's priority and its process' states, fragments and its life cycle.

Unit 3

User Interface Design: Layouts, optimizing layout hierarchies, form widgets, text fields, button control, toggle buttons, spinners, images, menu, dialog.

Unit 4

Broadcast receivers, notifications and services: Broadcast sender, receiver, broadcasting events with intents, listening for broadcasts with broadcast receivers, broadcasting ordered intents, broadcasting sticky intents, pending intents, creating notifications, setting and customizing the notification tray UI. Create, start, and stop services, binding services to activities, using asynctasks to manage background processing, handler, looper and runnable

Unit 5

Database and Content provider: SQLite, Content Values and Cursors, creating SQLite databases, querying a database, adding, updating, and removing rows, Creating Content Providers, implement content provider's queries and its usage.

Practical

1. Create "Hello World" application. That will display "Hello World" in the middle of the screen in the emulator. Also display "Hello World" in the middle of the screen in the Android Phone.
2. Create an application with three buttons (increment, decrement and reset) and a textview aligned vertically. On clicking, increment/decrement button, the value of the textview should increment/decrement by 1 while selecting reset button, the value of textview should become zero.
3. Create an application with login module. (Check username and password).
4. Create spinner with strings taken from resource folder (res >> value folder) and on changing the spinner value, Image will change.
5. Create a menu with 5 options and selected option should appear in text box.
6. Create a list of all courses in your college and on selecting a particular course teacher-in-charge of that course should appear at the bottom of the screen.
7. Create an application with three option buttons, on selecting a button colour of the screen will change.
8. Create an application to display various activity life cycle and fragment life cycle methods.
9. Create an application with 2 fragments, one to set the background and other to set the fore-color of the text.
10. Create an application with an activity having EditText and a button (with name "Send"). On clicking Send button, make use of implicit intent that uses a Send Action and let user select app from app chooser and navigate to that application.
11. Create a Login application. On successful login, use explicit intent to second activity displaying welcome message (Welcome Username) to the user and a logout button. When user presses logout button, a dialog box with a message ("Are you sure you want to exit?") and two buttons ("Yes" and "No") should appear to confirm logout. On "Yes" button click, go to login activity and on "No", stay on the same activity.
12. Create an application for Broadcast sender and receivers.
13. Create an application to create notification having icon, text and title.

14. Create an application to create services.
15. Create an application to Create, Insert, update, Delete and retrieve operation on the database.

References

1. Griffiths, D., & Griffiths, D., (2015). *Head First Android Development*, O'reilly Media.
2. Meier, R.,(2012). *Professional Android™ 4 Application Development*. John Wiley & Sons, Inc.

Additional Resources:

1. Murphy, M. L. (2018). *The Busy Coder's Guide to Android Development*, CommonsWare
2. Phillips, B., Stewart, C., Hardy, B. & Marsicano, K. (2015). *Android Programming: The Big Nerd Ranch Guide*.Big Nerd Ranch. Guides.
3. Sheusi, J. C. (2013). *Android Application Development for Java Programmers*. Cengage Learning.

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1-2	Introduction: Review to JAVA & OOPS Concepts
3	History of Android, Introduction to Android Operating Systems, Android Development Tools
4	Android Architecture Android components including activities, view and view group, services, content providers, broadcast receivers, intents, parcels, instance state.
5-6	User Interface Architecture, Application context, explicit intents, returning results from activities, implicit intents, intent filter and intent resolution, and applications of implicit intents
7	Activity life cycle, activity stack, application's priority and its process' states, fragments and its life cycle.

8	User Interface Design: Layouts, optimizing layout hierarchies,
9-10	Widgets with event handling: TextView, button control, toggle buttons, spinners, images, menu, dialog.
11	Broadcast sender and receivers: Broadcast sender, receiver, broadcasting events with intents, listening for broadcasts with broadcast receivers, broadcasting ordered intents, broadcasting sticky intents,
12	Notifications: pending intents, creating notifications, setting and customizing the notification tray UI.
13	Services: Create, start, and stop services, binding services to activities, using async tasks to manage background processing, handler, looper and runnable
14-15	Database and Content provider: SQLite, Content Values and Cursors, creating SQLite databases, querying a database, adding, updating, and removing rows, Creating Content Providers, implement content provider's queries and its usage.

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class.

Keywords

Android App Development, Activities, Fragments, User interfaces, Intents, Broadcast sender/receivers, Services, Notifications, SQLite Database

Introduction to R Programming (BHCS20B) Skill-Enhancement Elective Course - (SEC)

Credit: 04

Course Objective

This course introduces R, which is a popular statistical programming language. The course covers data reading and its manipulation using R, which is widely used for data analysis

internationally. The course also covers different control structures and design of user-defined functions. Loading, installing and building packages are covered.

Course Learning Outcomes

On successful completion of the course, students will be able to do following:

1. Develop an R script and execute it
2. Install, load and deploy the required packages, and build new packages for sharing and reusability
3. Extract data from different sources using API and use it for data analysis
4. Visualize and summarize the data
5. Design application with database connectivity for data analysis

Detailed Syllabus

Unit 1

Introduction: R interpreter, Introduction to major R data structures like vectors, matrices, arrays, list and data frames, Control Structures, vectorized if and multiple selection, functions.

Unit 2

Installing, loading and using packages: Read/write data from/in files, extracting data from web-sites, Clean data, Transform data by sorting, adding/removing new/existing columns, centring, scaling and normalizing the data values, converting types of values, using string in-built functions, Statistical analysis of data for summarizing and understanding data, Visualizing data using scatter plot, line plot, bar chart, histogram and box plot

Unit 3

Designing GUI: Building interactive application and connecting it with database.

Unit 4

Building Packages.

Practical

Q1. Write an R script to do the following:

- a) simulate a sample of 100 random data points from a normal distribution with mean 100 and standard deviation 5 and store the result in a vector.
- b) visualize the vector created above using different plots.
- c) test the hypothesis that the mean equals 100.
- d) use wilcox test to test the hypothesis that mean equals 90.

Q2. Using the Algae data set from package DMwR to complete the following tasks.

- a) create a graph that you find adequate to show the distribution of the values of algae a6.
- b) show the distribution of the values of size 3.
- c) check visually if oPO4 follows a normal distribution.
- d) produce a graph that allows you to understand how the values of NO3 are distributed across the sizes of river.
- e) using a graph check if the distribution of algae a1 varies with the speed of the river.
- f) visualize the relationship between the frequencies of algae a1 and a6. Give the appropriate graph title, x-axis and y-axis title.

Q3. Read the file Coweeta.CSV and write an R script to do the following:

- a) count the number of observations per species.
- b) take a subset of the data including only those species with at least 10 observations.
- c) make a scatter plot of biomass versus height, with the symbol colour varying by species, and use filled squares for the symbols. Also add a title to the plot, in italics.
- d) log-transform biomass, and redraw the plot.

Q4. The built-in data set mammals contain data on body weight versus brain weight. Write R commands to:

- a) Find the Pearson and Spearman correlation coefficients. Are they similar?
- b) Plot the data using the plot command .
- c) Plot the logarithm (log) of each variable and see if that makes a difference.

Q5. In the library MASS is a dataset UScereal which contains information about popular breakfast cereals. Attach the data set and use different kinds of plots to investigate the following relationships:

- a) relationship between manufacturer and shelf
- b) relationship between fat and vitamins
- c) relationship between fat and shelf
- d) relationship between carbohydrates and sugars
- e) relationship between fibre and manufacturer
- f) relationship between sodium and sugars

Q6. Write R script to:

- a) Do two simulations of a binomial number with $n = 100$ and $p = .5$. Do you get the same results each time? What is different? What is similar?

- b) Do a simulation of the normal two times. Once with $n = 10$, $\mu = 10$ and $\sigma = 10$, the other with $n = 10$, $\mu = 100$ and $\sigma = 100$. How are they different? How are they similar? Are both approximately normal?

Q7. Create a database medicines that contains the details about medicines such as {manufacturer, composition, price}. Create an interactive application using which the user can find an alternative to a given medicine with the same composition.

Q8. Create a database songs that contains the fields {song_name, mood, online_link_play_song}. Create an application where the mood of the user is given as input and the list of songs corresponding to that mood appears as the output. The user can listen to any song from the list via the online link given.

Q9. Create a package in R to perform certain basic statistics functions.

Mini project using data set of your choice from Open Data Portal (<https://data.gov.in/>) for the following exercises

References

1. Cotton, R., *Learning R: a step by step function guide to data analysis*. 1st edition. O'reilly Media Inc.

Additional Resources:

2. Gardener, M.(2017). *Beginning R: The statistical programming language*, WILEY.

3. Lawrence, M., & Verzani, J. (2016). *Programming Graphical User Interfaces in R*. CRC press. (ebook)

Web Resources

<https://jrnold.github.io/r4ds-exercise-solutions/index.html>

<https://www.r-project.org/>

<https://cran.r-project.org/>

Course Teaching Learning Process

- Use of ICT tools in conjunction with traditional class room teaching methods
- Interactive sessions
- Class discussions

Tentative weekly teaching plan is as follows:

Week	Content
1	R interpreter, Introduction to major R data structures like vectors, matrices, arrays, list and data frames
2	Flow control and loops, looping over list and array
3	User-defined functions
4	Installing, loading different packages for file handling
5	Reading and writing files of different formats using inbuilt packages
6	Using inbuilt packages for data cleaning
7	Transformation of data for statistical analysis
8	Exploring and summarizing data using statistical methods: mean, median, mode
9	Exploring and summarizing data using statistical methods: quantiles, Building contingency table
10	Data visualization using Scatter Plot, line graph, histogram, barchart, boxplot
11	Designing GUI
12	Continuing with creating GUI for application, building package
13-14	Using inbuilt packages for database connectivity
15	Building complete application with GUI and database connectivity

Assessment Methods

Written tests, assignments, quizzes, presentations as announced by the instructor in the class

Keywords

R data structures, flow control, packages, functions

Acknowledgement (in alphabetical order)

Ajay Jaiswal	Shaheed Sukhdev College of Business Studies
Alka Khurana	Hansraj College
Anamika Gupta	Shaheed Sukhdev College of Business Studies
Anjali Thukral	Keshav Mahavidyalaya
Anuja Soni	Deen Dayal Upadhyay College
Anuradha Khattar	Miranda House
Archana Singhal	IP College
Arpita Sharma	Deen Dayal Upadhyaya College
Bharti	Hans Raj College
Bhavna Gupta	Keshav Mahavidyalaya
Chanderkant Samal	AND College
Charu Puri	PGDAV College
Deepti Chopra	IP college for women
Divya Kawatra	Hansraj College
Harita Ahuja	Acharya Narendra Dev College
Harmeet Kaur	Hansraj College
Hema Banati	Dayal Singh College
Manisha Bansal	Indraprastha College for Women, University of Delhi
Manju Bala	IP College
Manju Bhardwaj	Maitreyi College
Maulein Pathak	Keshav Mahavidyalaya, University of Delhi
Megha Khandelwal	Department of Computer Science
Ms Vandana Kalra	Shri Guru Gobind Singh College of Commerce
Ms. Bharti Kumar	Shyam Lal College (Eve.)
Nagendra Duhan	Bhagni Nivedita College
Naveen Kumar	Deptt. of Computer Science
Neelima Gupta	Deptt. of Computer Science
Neeraj Kumar Sharma	Ram Lal Anand College
Nidhi Arora	Kalindi College
Nikhil Rajput	Ramanujan College

Nisha	Deptt. of Computer Science
Onkar Singh	Shaheed Sukhdev College of Business Studies
PK Hazra	Deptt. of Computer Science
Preeti Marwaha	Acharya Narendra Dev College
Priti Sehgal	Keshav Mahavidyalaya
Priyanka Rathi	Deptt. of Computer Science
Punam Bedi	Deptt. of Computer Science
Rajan Gupta	DDU college
Rajni Bala	Deen Dayal Upadhyay College
Rakhi Saxena	Deshbandhu College
Rampal Rana	Deen Dayal Upadhyay College
Ravi Yadav	Keshav Mahavidyalaya, University of Delhi
Ravish Sharma	PGDAV College
Reena Kasana	Deptt. of Computer Science
Rochana Chaturvedi	Keshav Mahavidyalaya
Roli Bansal	Keshav Mahavidyalaya
Ronnie Chakre	Deptt. of Computer Science
S K Muttoo	Deptt. of Computer Science
Sahil Pathak	Ramanujan College
Sameer Anand	Shaheed Sukhdev College of Business Studies
Sangeeta Srivastava	BCAS
Sapna Varshney	Deptt. of Computer Science
Sanjeet Kumar	DDU College
Sarabjeet Kochhar	Indraprastha College for Women
Seema Aggarwal	Miranda House
Sharanjit Kaur	Acharya Narendra Dev College
Shikha Badhani	Maitri College
Shikha Gupta	Shaheed Sukhdev College of Business Studies
Shikha Verma	Ram Lal Anand College
Sonika Thakral	Shaheed Sukhdev College of Business Studies
Sujata Khatri	Deen Dayal Upadhyay College
	Shaheed rajguru college of applied science for women
Suruchi Chawla	LSR College
Sushila Madan	Sri Guru Gobind Singh College of Commerce
Vandana Kalra	Deptt. of Computer Science
Vasudha Bhatnagar	Acharya Narendra Dev College
Vibha Gaur	Indraprastha College for Women
Vimala Parihar	Keshav Mahavidyalaya
Vinita Jindal	