BUN
ISZ

7. 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:
   CLA
   CMA
   CME
   HLT
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
   INC
   SPA
   SNA
   SZE

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:
   CIR
   CIL \
10. 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).
11. Write an assembly program that reads in integers and adds them together; until zero is read in. Then it outputs the sum.

Note: Examination scheme and mode shall be as prescribed by the Examination Branch, University of Delhi, from time to time.

DSC07/DSC02/GE4a: DATA STRUCTURES

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

	Credits	Credit distribution of the course		

Course title & Code		Lecture	Tutorial	Practical /Practice	Eligibility criteria	Pre-requisite of the course (if any)
Data Structures	**4**	**3**	**0**	**1**	**Passed 12th class with Mathematics**	Programming using C++

Course Objectives

The course aims at developing the ability to use basic data structures like arrays, stacks, queues, lists, and trees to solve problems. C++ is chosen as the language to implement the implementation of these data structures.

Learning Outcomes

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

● Compare two functions for their rates of growth.

● Understand abstract specification of data-structures and their implementation.

● Compute time and space complexity of operations on a data-structure.

● Identify the appropriate data structure(s) for a given application and understand the trade-offs involved in terms of time and space complexity.

● Apply recursive techniques to solve problems.

Syllabus

Unit-1 **(9 hours)**

Growth of Functions, Recurrence Relations: Functions used in analysis, asymptotic notations, asymptotic analysis, recurrence, Master Theorem.

Unit-2 **(16 hours)**

Arrays, Linked Lists, Stacks, Queues: Arrays: array operations, applications, two dimensional arrays, dynamic allocation of arrays; Linked Lists: singly linked lists, doubly linked lists, circularly linked lists, Stacks: stack as an ADT, implementing stacks using arrays, implementing stacks using

linked lists, applications of stacks; Queues: queue as an ADT, implementing queues using arrays, implementing queues using linked lists,. Time complexity analysis.

Unit-3 **(5 hours)**

Recursion: Recursive functions, linear recursion, binary recursion.

Unit-4 **(6 hours)**

Trees, Binary Trees: Trees: definition and properties, tree traversal algorithms, and their time complexity analysis; binary trees: definition and properties, traversal of binary trees, and their time complexity analysis.

Unit-5 **(7 hours)**

Binary Search Trees: Binary Search Trees: insert, delete, search operations, time complexity analysis of these operations

Unit-6 **(2 hours)**

Binary Heap: Binary Heaps: heaps, heap operations.

Essential/recommended readings

1. Goodrich, M.T., Tamassia, R., & Mount, D., Data Structures and Algorithms Analysis in C++, 2nd edition, Wiley, 2011. 4 th
2. Cormen, T.H., Leiserson, C.E., Rivest, R. L., Stein C. Introduction to Algorithms, edition, Prentice Hall of India, 2022. Additional references

Additional References

1. Sahni, S. Data Structures, Algorithms and applications in C++, 2nd edition, Universities Press, 2011.
2. Langsam Y., Augenstein, M. J., & Tanenbaum, A. M. Data Structures Using C and C++, Pearson, 2009.

Practical List **(30 hours)**

1. Write a program to implement singly linked list as an ADT that supports the following operations:
   a. Insert an element x at the beginning of the singly linked list
   b. Insert an element x at ith position in the singly linked list
   c. Remove an element from the beginning of the singly linked list
   d. Remove an element from ith position in the singly link

e. Search for an element x in the singly linked list and return its pointer
f. Concatenate two singly linked lists

2. Write a program to implement doubly linked list as an ADT that supports the following operations:
   a. Insert an element x at the beginning of the doubly linked list
   b. Insert an element x at ith position in the doubly linked list
   c. Insert an element x at the end of the doubly linked list
   d. Remove an element from the beginning of the doubly linked list
   e. Remove an element from ith position in the doubly linked list.
   f. Remove an element from the end of the doubly linked list
   g. Search for an element x in the doubly linked list and return its pointer (viii) Concatenate two doubly linked lists
3. Write a program to implement circular linked list as an ADT which supports the following operations:
   a. Insert an element x at the front of the circularly linked list
   b. Insert an element x after an element y in the circularly linked list
   c. Insert an element x at the back of the circularly linked list
   d. Remove an element from the back of the circularly linked list
   e. Remove an element from the front of the circularly linked list
   f. Remove the element x from the circularly linked list
   g. Search for an element x in the circularly linked list and return its pointer
   h. Concatenate two circularly linked lists
4. Implement a stack as an ADT using Arrays.
5. Implement a stack as an ADT using the Linked List ADT.
6. Write a program to evaluate a prefix/postfix expression using stacks.
7. Implement Queue as an ADT using the circular Arrays.
8. Implement Queue as an ADT using the Circular Linked List ADT.
9. Write a program to implement Binary Search Tree as an ADT having the following operations:
   a. Insert an element x
   b. Delete an element x
   c. Search for an element x in the BST and change its value to y and then place the node with value y at its appropriate position in the BST
   d. Display the elements of the BST in preorder, inorder, and postorder traversal
   e. Display the elements of the BST in level-by-level traversal
   f. Display the height of the BST

Note: Examination scheme and mode shall be as prescribed by the Examination Branch, University of Delhi, from time to time.

DSC08/DSC04/GE5a: OPERATING SYSTEMS

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical / Practice		
Operating Systems	4	3	0	1	Passed 12th class with Mathematics	Programming using C++ / Python/Java

Course Objectives

The course provides concepts that underlie all operating systems and are not tied to any particular operating system. The emphasis is on explaining the need and structure of an operating system using its common services such as process management (creation, termination etc.), CPU Scheduling, Process Synchronization, Handling Deadlocks, main memory management, virtual memory, secondary memory management. The course also introduces various scheduling algorithms, structures, and techniques used by operating systems to provide these services.

Learning Outcomes

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

● Describe the need of an operating system and define multiprogramming and Multithreading concepts.
● Implement the process synchronization service (Critical Section, Semaphores), CPU scheduling service with various algorithms.
● Implement Main memory Management (Paging, Segmentation) algorithms, Handling of Deadlocks
● Identify and appreciate the File systems Services, Disk Scheduling service

Syllabus

Unit-1 **(6 hours)**

Introduction: Operating Systems (OS) definition and its purpose, OS Structure, OS Operations: Dual and Multi-mode, OS as resource manager.

Unit-2 **(9 hours)**

Operating System Structures: OS Services, System Calls: Process Control, File Management, Device Management, and Information Maintenance, Inter-process Communication, and Protection, System programs, OS structure- Simple, Layered, Microkernel, and Modular.

Unit-3 **(10 hours)**

Process Management: Process Concept, States, Process Control Block, Process Scheduling, Schedulers, Context Switch, Operation on processes, Threads, Multicore Programming, Multithreading Models, Process Scheduling Algorithms: First Come First Served, Shortest-Job-First, Priority & Round-Robin, Process Synchronization: The critical section problem, Deadlock characterization, Deadlock handling.

Unit-4 **(11 hours)**

Memory Management: Physical and Logical address space, Swapping, Contiguous memory allocation strategies - fixed and variable partitions, Segmentation, Paging. Virtual Memory Management: Demand Paging and Page Replacement algorithms: FIFO Page Replacement, Optimal Page replacement, LRU page replacement.

Unit-5 **(9 hours)**

File System: File Concepts, File Attributes, File Access Methods, Directory Structure: Single Level, Two-Level, Tree-Structured, and Acyclic-Graph Directories. Mass Storage Structure: Magnetic Disks, Solid-State Disks, Magnetic Tapes, Disk Scheduling algorithms: FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOk Scheduling.

Essential/recommended readings

1. Silberschatz, A., Galvin, P. B., Gagne G. Operating System Concepts, 9 th edition, John Wiley Publications, 2016.

2. Tanenbaum, A. S. Modern Operating Systems, 3 rd edition, Pearson Education, 2007.
3. Stallings, W. Operating Systems: Internals and Design Principles, 9 th edition, Pearson Education, 2018.

Additional References

1. Dhamdhere, D. M., Operating Systems: A Concept-based Approach, 2nd edition, Tata McGraw-Hill Education, 2017.
2. Kernighan, B. W., Rob Pike, R. The Unix Programming Environment, Englewood Cliffs, NJ: Prentice-Hall, 1984.

Practicals

1. Execute various Linux commands for:
   a. Information Maintenance: wc, clear, cal, who, date, pwd
   b. File Management: cat, cp, rm, mv, cmp, comm, diff, find, grep, awk
   c. Directory Management : cd, mkdir, rmdir, ls
2. Execute various Linux commands for:
   a. Process Control: fork, getpid, ps, kill, sleep
   b. Communication: Input-output redirection, Pipe
   c. Protection Management: chmod, chown, chgrp
3. Write a programme (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.
4. Write a program to to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information).
5. Write a program to report behaviour of Linux kernel including information on 19 configured memory, amount of free and used memory. (Memory information)
6. Write a program to copy files using system calls.
7. Use an operating system simulator to simulate operating system tasks.
8. Write a program to implement scheduling algorithms FCFS/ SJF/ SRTF/ non preemptive scheduling algorithms.
9. Write a program to calculate the sum of n numbers using Pthreads. A list of n numbers is divided into two smaller lists of equal size, and two separate threads are used to sum the sublists.
10. Write a program to implement first-fit, best-fit and worst-fit allocation strategies.

Note: Examination scheme and mode shall be as prescribed by the Examination Branch, University of Delhi, from time to time.

9. SVM classification

10. K-Means Clustering

11. Hierarchical Clustering

DSC 9 / DSC16/GE6e/DSE: ARTIFICIAL INTELLIGENCE

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course
		Lecture	Tutorial	Practical/ Practice		
Artificial Intelligence	4	3	0	1	Pass in Class XII	Programming using C++/Programming using Python/Object Oriented Programming using Python

Course Objectives

The objectives of this course are to:

- To introduce basic concepts and techniques of Artificial Intelligence (AI).
- To apply informed search techniques for different applications.
- To learn various knowledge representation techniques and writing Prolog programs.
- To learn about the latest techniques for developing AI systems.

Learning outcomes

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

- identify problems that are amenable to solutions by specific AI methods.
- state the utility of different types of AI agents.
- apply different informed search techniques for solving problems.
- use knowledge representation techniques for AI systems.

SYLLABUS

Unit 1 **6 Hours**

Introduction: Introduction to artificial intelligence, background and applications, Turing test, Weak AI, Strong AI, Narrow AI, Artificial General Intelligence, Super AI, rational agent approaches to AI, introduction to intelligent agents, their structure, behavior and task environment.

Unit 2 **12 Hours**

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

Unit 3 **16 Hours**

Knowledge Representation: Propositional logic, First-Order Predicate logic, resolution principle, unification, semantic nets, conceptual dependencies, frames, and scripts, production rules, Introduction to Programming in Logic (PROLOG).

Unit 4 **8 Hours**

Understanding Natural Languages: Components and steps of communication, the contrast between formal and natural languages in the context of grammar, Chomsky hierarchy of grammars, parsing, and semantics, Parsing Techniques, Context-Free and Transformational Grammars, Recursive transition nets.

Unit 5 **3 Hours**

AI The Present and the Future: Symbolic AI, Data-driven AI and Machine Learning, Introduction to Machine Learning and Deep Learning based AI, Interpretable and Explainable AI, Ethics of AI: benefits and risks of AI.

Essential/recommended readings

1. Russell, Stuart, J. and Norvig, Peter, *Artificial Intelligence - A Modern Approach*, Pearson, 4th edition, 2020..
2. Bratko, Ivan, *Prolog Programming for Artificial Intelligence*, Addison-Wesley, Pearson Education, 4th edition, 2012.
3. Patterson, DAN,W, *Introduction to A.I. and Expert Systems* – PHI, 2007.
4. Clocksin, W., F. and Mellish, *Programming in PROLOG*, 5th edition, Springer, 2003.

Additional references

1. Kaushik, Saroj, *Artificial Intelligence*, Cengage Learning India, 2011.
2. Rich, Elaine and Knight, Kelvin, *Artificial Intelligence*, 3rd edition, Tata McGraw Hill, 2010

Practical List :

Practical exercises such as

1. Write a program in Prolog to implement TowerOfHanoi(N) where N represents the number of disks.
2. Write a program to implement the Hill climbing search algorithm in Prolog.
3. Write a program to implement the Best first search algorithm in Prolog.
4. Write a program to implement A* search algorithm in Prolog.
5. Write a program to implement the min-max search algorithm in Prolog.
6. Write a program to solve the Water-Jug Problem in Prolog.
7. Implement sudoku problem (minimum 9×9 size) using constraint satisfaction in Prolog.
8. Write a Prolog program to implement the family tree and demonstrate the family relationship.
9. Write a Prolog program to implement knowledge representation using frames with appropriate examples.
10. 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.
11. Write a Prolog program to implement reverse(L, R) where List L is original and List R is reversed list.
12. Write a Prolog program to generate a parse tree of a given sentence in English language assuming the grammar required for parsing.
13. Write a Prolog program to recognize context free grammar $a^n b^n$.