UNIVERSITY OF DELHI

CNC-II/093/1(25)/2023-24/64
Dated: 30.05.2023

NOTIFICATION

Sub: Amendment to Ordinance V

[E.C Resolution No. 60/ (60-1-7/) dated 03.02.2023]

Following addition be made to Appendix-II-A to the Ordinance V (2-A) of the Ordinances of the University;

Add the following:

Syllabi of Semester-III of the following departments under Faculty of Mathematical Sciences based on Under Graduate Curriculum Framework -2022 implemented from the Academic Year 2022-23.

FACULTY OF MATHEMATICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

BSC. (HONS.) COMPUTER SCIENCE

DISCIPLINE SPECIFIC CORE COURSE -7 (DSC-7) : Data Structures

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		
DSC07 Data Structures	4	3	0	1	Passed 12th class with Mathematics	Programming using Python/Object Oriented Programming with C++

Learning 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 OF DSC-7

Unit 1 (9 hours)

Growth of Functions, Recurrence Relations: Functions used in analysis, asymptotic notations, asymptotic analysis, solving recurrences using recursion trees, 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, Balanced Search Trees: Binary Search Trees: insert, delete, search operations, time complexity analysis of these operations; Balanced Search Trees: insert, search operations, time complexity analysis of these operations. Time complexity analysis.

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.
2. Cormen, T.H., Leiserson, C.E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 4th edition, Prentice Hall of India, 2022.

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 (If any): (30 Hours)

Practical exercises such as

1. Write a program to implement singly linked list as an ADT that supports the following operations:
   (i) Insert an element x at the beginning of the singly linked list
   (ii) Insert an element x at *i*th position in the singly linked list
   (iii) Remove an element from the beginning of the singly linked list
   (iv) Remove an element from *i*th position in the singly link
   (v) Search for an element x in the singly linked list and return its pointer
   (vi) Concatenate two singly linked lists

2. Write a program to implement doubly linked list as an ADT that supports the following operations:
   (i) Insert an element x at the beginning of the doubly linked list
   (ii) Insert an element x at *i*th position in the doubly linked list
   (iii) Insert an element x at the end of the doubly linked list
   (iv) Remove an element from the beginning of the doubly linked list
   (v) Remove an element from *i*th position in the doubly linked list.
   (vi) Remove an element from the end of the doubly linked list
   (vii) 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:
   (i) Insert an element x at the front of the circularly linked list
   (ii) Insert an element x after an element y in the circularly linked list
   (iii) Insert an element x at the back of the circularly linked list
   (iv) Remove an element from the back of the circularly linked list
   (v) Remove an element from the front of the circularly linked list
   (vi) Remove the element x from the circularly linked list
   (vii) Search for an element x in the circularly linked list and return its pointer

(viii) 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 which supports the following operations:
   (i) Insert an element x
   (ii) Delete an element x
   (iii) 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
   (iv) Display the elements of the BST in preorder, inorder, and postorder traversal
   (v) Display the elements of the BST in level-by-level traversal
   (vi) Display the height of the BST

10. Write a program to implement a balanced search tree as an ADT.

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

DISCIPLINE SPECIFIC CORE COURSE – 8 (DSC-8): Operating Systems

Credit distribution, Eligibility and Prerequisites 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**		
DSC 08 Operating Systems	**4**	**3**	**0**	**1**	**Passed 12th class with Mathematics**	Programming using Python/Object Oriented Programming with C++, Computer System Architecture

Learning 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 OF DSC-8

Unit 1 (6 hours)

Introduction: Operating Systems (OS) definition and its purpose, Multiprogrammed and Time Sharing Systems, 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, PThreads, Process Scheduling Algorithms: First Come First Served, Shortest-Job-First, Priority & Round-Robin, Process Synchronization: The critical-section problem and Peterson's Solution, 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,* 9th edition, John Wiley Publications, 2016.
2. Tanenbaum, A. S. *Modern Operating Systems*, 3rd edition, Pearson Education, 2007.
3. Stallings, W. *Operating Systems: Internals and Design Principles,* 9th 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.

Suggested Practical List (If any): (30 Hours)

Practical exercises such as

1. Execute various Linux commands for:
   i. Information Maintenance: wc, clear, cal, who, date, pwd
   ii. File Management: cat, cp, rm, mv, cmp, comm, diff, find, grep, awk
   iii. Directory Management : cd, mkdir, rmdir, ls
2. Execute various Linux commands for:
   i. Process Control: fork, getpid, ps, kill, sleep
   ii. Communication: Input-output redirection, Pipe
   iii. Protection Management: chmod, chown, chgrp
3. Write a programme (using fork() and/or exec() commands) where parent and child execute:
   i. same program, same code.
   ii. same program, different code.
   iii. 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.

DISCIPLINE SPECIFIC CORE COURSE– 9 (DSC-9): Numerical Optimization

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**		
DSC09 Numerical Optimization	**4**	**3**	**0**	**1**	**Passed 12th class with Mathematics**	Programming using Python/Object Oriented Programming with C++

Learning Objectives

The course aims to provide students with the experience of mathematically formulating a large variety of optimization/decision problems emerging out of various fields like data science, machine learning, business, and finance. The course focuses on learning techniques to optimize problems in order to obtain the best possible solution.

Learning outcomes

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

- Mathematically formulate the optimization problems using the required number of independent variables.
- Define constraint functions on a problem.
- Check the feasibility and optimality of a solution.
- Apply conjugate gradient method to solve the problem.

SYLLABUS OF DSC-9

Unit 1 (6 hours)

Introduction: Mathematical Formulation using example, Continuous versus Discrete Optimization, Constrained and Unconstrained Optimization, Global and Local Optimization, Stochastic and Deterministic Optimization, Convexity, Optimization Algorithms

Unit 2 (14 hours)

Fundamentals of Unconstrained Optimization: Concept of a Solution - Recognizing a Local Minimum, Nonsmooth Problems, Overview of Algorithms - Two Strategies: Line Search and Trust Region, Search Directions for Line Search Methods, Models for Trust-Region Methods, Scaling. Line Search - Convergence of Line Search Methods, Rate of Convergence - Convergence Rate of Steepest Descent; Newton's Method, Quasi-Newton Methods. Trust Region - The Cauchy Point Algorithm; Global Convergence - Reduction Obtained by the Cauchy Point; Convergence to Stationary Points.

Unit 3 (7 hours)

Conjugate Gradient Methods: Basic Properties of the Conjugate Gradient Method, A Practical Form of the Conjugate Gradient Method, and Rate of Convergence

Unit 4 (8 hours)

Calculating Derivatives: Finite-Difference Derivative Approximations, Approximating the Gradient, Approximating a Sparse Jacobian, Approximating the Hessian, Approximating a Sparse Hessian

Unit 5 (10 hours)

Theory of Constrained Optimization: Local and Global Solutions, Smoothness, Examples - A Single Equality Constraint, A Single Inequality Constraint, Two Inequality Constraints, Tangent Cone and Constraint Qualifications, First-Order Optimality Condition, Second-Order Conditions - Second-Order Conditions and Projected Hessians. Linear and non-linear constrained optimization, augmented Lagrangian Method

Essential/recommended readings

1. J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd edition, Springer Series in Operations Research, 2006.
2. A, Mehra, S Chandra, Jayadeva, *Numerical Optimization with Applications*, Narosa Publishing House, New Delhi, 2009,

Additional References

1. R. W. Hamming, *Numerical Methods for Scientists and Engineers,* 2nd edition, Dover Publications, 1986.
2. Q. Kong, T. Siauw, A. Bayen, *Python Programming and Numerical Methods: A Guide for Engineers and Scientists,* 1st edition, 2020.

Suggested Practical List (If any) :(30 Hours)

Practical exercises such as

Write a program to implement the following methods:

Constrained and Unconstrained Optimization, Global and Local Optimization, Line Search and Trust Region, Convergence of Line Search Methods, Rate of Convergence - Convergence Rate of Steepest Descent, Newton's Method, Quasi-Newton Methods, The Cauchy Point algorithm, Finite-Difference Derivative Approximations, Convergence to Stationary Points, Conjugate Gradient Method, Rate of Convergence, Approximating a Sparse Jacobian, Approximating the Hessian, Approximating a Sparse Hessian, First-Order Optimality Condition, Second-Order Conditions - Second-Order Conditions, and Projected Hessians. Linear and non-linear constrained optimization Augmented Lagrangian Methods.

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