COURSEPACK (Fall 2024-25)

1. SCHEME

The scheme is an overview of work-integrated learning opportunities and gets students out into the real world. This will show what a course entails.

<table>
<tr><td>Course Title</td><td colspan="3">Advanced Data Structures and Algorithms</td><td colspan="3">Course Type</td><td colspan="3">Integrated</td></tr>
<tr><td>Course Code</td><td colspan="3">R1UC503B</td><td colspan="3">Class</td><td colspan="3">B. Tech. (CSE) V Sem.</td></tr>
<tr><td rowspan="6">Instruction delivery</td><td>Activity</td><td>Credits</td><td>Weekly Hours</td><td colspan="4" rowspan="2">Total Number of Classes per Semester</td><td colspan="2" rowspan="2">Assessment in Weightage</td></tr>
<tr><td>Lecture</td><td>3</td><td>4</td></tr>
<tr><td>Tutorial</td><td>0</td><td>0</td><td rowspan="3">Theory</td><td rowspan="3">Tutorial</td><td rowspan="3">Practical</td><td rowspan="3">Self-study</td><td rowspan="3">CIE</td><td rowspan="3">SEE</td></tr>
<tr><td>Practical</td><td>1</td><td>2</td></tr>
<tr><td>Self-study</td><td>0</td><td>0</td></tr>
<tr><td>Total</td><td>4</td><td>8</td><td>40</td><td>0</td><td>40</td><td></td><td>50%</td><td>50%</td></tr>
<tr><td>Course Lead</td><td colspan="2">Dr. Anupam Kumar Sharma</td><td>Course Coordinator</td><td colspan="6">Dr. Subhash Gupta</td></tr>
<tr><td>Names Course Instructors</td><td colspan="3">Theory
Dr. Anupam Kumar Sharm
Dr. Subhash Chandra Gupta
Dr. Manoj Kumar Tyagi
Dr. S Karpaga Selvi
Dr. Saurabh Singh
Dr. Gaurav Agrawal
Dr. Pooja Singh
Dr. Sahil Kansal
Dr. Akhilendra Kumar Khare
Mr. Akhilesh Kumar Tripathi
Mr. Avaneesh Kumar Yadav
Mr. Gopal Chandra Jana
Mr. Harshit Jain
Dr. Himanshu Sharma
Mr. Neeraj Kumar Bharti
Ms. Pragya Agarwal
Ms. Shweta
Dr. Vipul Narayan
Mr. Mayank Choudhary
Mr. Jitendra
Dr. Mohammad Faiz
Mr. Vinod Kumar
Mr. Vikas Kumar
Mr. Ashwin Perti</td><td colspan="6">Practical
Dr. Anupam Kumar Sharma
Dr. Subhash Chandra Gupta
Dr. Manoj Kumar Tyagi
Dr. S Karpaga Selvi
Dr. Saurabh Singh
Dr. Gaurav Agrawal
Dr. Pooja Singh
Dr. Sahil Kansal
Dr. Akhilendra Kumar Khare
Mr. Akhilesh Kumar Tripathi
Mr. Avaneesh Kumar Yadav
Mr. Gopal Chandra Jana
Mr. Harshit Jain
Dr. Himanshu Sharma
Mr. Neeraj Kumar Bharti
Ms. Pragya Agarwal
Ms. Shweta
Dr. Vipul Narayan
Mr. Mayank Choudhary
Mr. Jitendra
Dr. Mohammad Faiz
Mr. Vinod Kumar
Mr. Vikas Kumar
Mr. Ashwin Perti</td></tr>
</table>

	Mr. Dileep Kumar Kushwaha Ms. Pragati Gupta Mr. Amar Singh Mr. Manish Kumar Maurya Ms. Arpan Kumari Mr. Deepak Dr. Kanika Thakur	Mr. Dileep Kumar Kushwaha Ms. Pragati Gupta Mr. Amar Singh Mr. Manish Kumar Maurya Ms. Arpan Kumari Mr. Deepak Dr. Kanika Thakur

2. COURSE OVERVIEW

An advanced course in data structures and algorithms builds upon the foundational knowledge of basic data structures and algorithms. It delves deeper into more complex data structures and advanced algorithmic techniques. This course explores advanced data structures and algorithmic techniques used in computer science and software development. It focuses on the design, analysis, and implementation of data structures and algorithms for solving complex computational problems efficiently.

3. COURSE OBJECTIVE

This course aims to familiarize students with algorithmic design techniques of dynamic programming, greedy programming, and backtracking as well as advanced data structures AVL Tree, B+ tree, Quad-Tree, Octree, and Trie. They will be taught to compute time and space complexities of algorithms and implement them.

4. PREREQUISITE COURSE

PREREQUISITE COURSE REQUIRED	**YES**	
If, YES please fill in the Details.	**Prerequisite course code**	**Prerequisite course name**
		Data Structure and algorithms.

5. PROGRAM OUTCOMES (POs): AS DEFINED BY CONCERNED THE APEX BODIES

PO1 Computing Science knowledge: Apply the knowledge of mathematics, statistics, computing science, and information science fundamentals to the solution of complex computer application problems.

PO2 Problem analysis: Identify, formulate, review research literature, and analyze complex computing science problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and computer sciences.

PO3 Design/development of solutions: Design solutions for complex computing problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4 Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern computing science and IT tools including prediction and modeling to complex computing activities with an understanding of the limitations.

PO6 IT specialist and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to theprofessional computing science and information science practice.

PO7 Environment and sustainability: Understand the impact of professional computing science solutions in societal and environmental contexts, and demonstrate the knowledge of, and need forsustainable development.

PO8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of computing science practice.

PO9 Individual and teamwork: Function effectively as an individual, and as a member or leaderin diverse teams, and in multidisciplinary settings.

PO10 Communication: Communicate effectively on complex engineering activities with the IT analyst community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 Project management and finance: Demonstrate knowledge and understanding of the computing science and management principles and apply these to one's own work, as a memberand leader in a team, to manage projects and in multidisciplinary environments.

PO12 Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

6. **PROGRAMME SPECIFIC OUTCOME (PSO): The students of Computer Science and Engineering shall:**

PSO1: Have the ability to work with emerging technologies in computing requisite to Industry4.0.

PSO2: Demonstrate Engineering Practice learned through industry internship and research project to solve live problems in various domains.

7. COURSE CONTENT (THEORY+ PRACTICAL)

Contents (Syllabus)

Theory

Review of Algorithmic Complexity: Definition and significance of algorithmic complexity Time and space complexity, Big O notation, Omega, and Theta notations, analyzing basic algorithms, solve problems on basic time complexity analysis.

Arrays: Static vs. dynamic arrays. Find the Maximum and Minimum Elements in an Array, Reverse an Array, Find the Kth Smallest/Largest Element in an Array.

Linked lists: Implementation of Singly Linked Lists, doubly linked list, and circular linked list. Operations on Linked List. Insertion, Deletion, Traversal.

Stacks: Implementation of Stack using linked list, Application of stack: infix, Prefix and Postfix Expressions, infix to postfix expression, Evaluation of postfix expression.

Recursion- Principles of recursion, Tail recursion, problem solving using recursion, Fibonacci numbers, a^n (a raised to the power n), reverse of a string.

Queues: Implementation and operations on Queue using linked list. Priority queue (heap).

Hashing: Concept of Hashing & Collision resolution Techniques used in Hashing.

Tree: Linked List Representation, Binary Search Tree, Tree Traversal algorithms: In-order, Pre-order, and Post-order, Constructing Binary Tree from given Tree Traversal, Operation of Insertion, Deletion, Searching Modification of data in Binary Search.

Graph: Representations: Adjacency Matrices, Adjacency List, Graph Traversal: Depth First Search & Breadth First Search. Detect Cycle in an Undirected Graph, Connected components in an Undirected Graph.

Minimum Cost Spanning Trees: Prims and Kruskal algorithm. Shortest Path algorithm: Warshal Algorithm.

Dynamic Programming: Elements of dynamic programming, longest common subsequence, Coin change problem, 0/1 Knapsack.

Greedy Algorithms: Elements of the greedy strategy, Fractional knapsack, Activity Selection, Huffman coding, job sequencing Problem.

Back Tracking: Sum of subset, N queens' problem.

Probabilistic Data Structures: Introduction to Probabilistic Algorithms, Advantages and Trade-offs of Probabilistic Data Structures, Applications and Use Cases, Structure and Function of Bloom Filters, Hash Functions and Their Role, False Positives and Space Efficiency, Variants of Bloom Filters (e.g., Counting Bloom Filter).

S. No.	Practical
1	**Find the Maximum and Minimum Elements in an Array:** Write a function to find the maximum and minimum elements in an array.
2	**Reverse an Array:** Write a function to reverse an array in place.
3	**Find the Kth Smallest/Largest Element in an Array:** Write a function to find the Kth smallest or largest element in an array.
4	**Sort an Array of 0s, 1s, and 2s:** Given an array containing only 0s, 1s, and 2s, sort the array in linear time.
5	**Move All Zeroes to End of Array:** Write a function to move all zeroes in an array to the end while maintaining the relative order of other elements.
6	**Reverse a Linked List:** Write a function to reverse a singly linked list.
7	**Detect a Cycle in a Linked List:** Write a function to detect if a cycle exists in a linked list.
8	**Find the Middle of a Linked List:** Write a function to find the middle element of a linked list.
9	**Merge Two Sorted Linked Lists:** Write a function to merge two sorted linked lists into one sorted linked list.
10	**Remove Nth Node from End of List:** Write a function to remove the Nth node from the start/end of a linked list.
11	**Implement a Stack Using Arrays/Lists:** Write a function to implement a stack using an array or list with basic operations: push, pop, peek, and isEmpty.
12	**Implement a Stack Using Linked List:** Write a function to implement a stack using alinked list with basic operations: push, pop, peek, and isEmpty.
13	**Check for Balanced Parentheses:** Write a function to check if a string containing parentheses is balanced.
14	**Evaluate Postfix Expression:** Write a function to evaluate a given postfix expression.
15	**Next Greater Element:** Write a function to find the next greater element for each elementin an array.
16	**Implement a Queue Using Arrays/Lists:** Write a function to implement a queue using an array or list with basic operations: enqueue, dequeue, front, and isEmpty.
17	**Implement a Queue Using Linked List:** Write a function to implement a queue using a linked list with basic operations: enqueue, dequeue, front, and isEmpty.
18	**Implement a Circular Queue:** Write a function to implement a circular queue with basic operations: enqueue, dequeue, front, rear, and isEmpty.
19	**Generate Binary Numbers from 1 to N:** Write a function to generate binary numbers from1 to N using a queue.
20	**Implement a Queue Using Stacks:** Write a function to implement a queue using two stacks. (vice-versa)
21	**Implement a Binary Tree:** Write a class to implement a basic binary tree with insert,delete, and traversal operations.
22	**Inorder Traversal:** Write a function to perform inorder traversal of a binary tree.
23	**Preorder Traversal:** Write a function to perform preorder traversal of a binary tree.
24	**Postorder Traversal:** Write a function to perform postorder traversal of a binary tree.

25	**Level Order Traversal:** Write a function to perform level order traversal of a binary tree.
26	**Height of a Binary Tree:** Write a function to find the height of a binary tree.
27	**Diameter of a Binary Tree:** Write a function to find the diameter of a binary tree.
28	**Check if a Binary Tree is Balanced:** Write a function to check if a binary tree is height balanced.
29	**Lowest Common Ancestor:** Write a function to find the lowest common ancestor of two nodes in a binary tree.
30	**Implement Graph Using Adjacency List:** Write a class to implement a basic graph usingan adjacency list with methods to add vertices and edges.
31	**Breadth-First Search (BFS):** Write a function to perform BFS on a graph from a givenstart vertex.
32	**Depth-First Search (DFS):** Write a function to perform DFS on a graph from a given start vertex.
33	**Detect Cycle in an Undirected Graph:** Write a function to detect if there is a cycle in an undirected graph.
34	**Connected Components in an Undirected Graph:** Write a function to find the number of connected components in an undirected graph.
35	**Find MST Using Kruskal's Algorithm:** Write a function to find the Minimum Spanning Tree of a graph using Kruskal's algorithm.
36	**Find MST Using Prim's Algorithm:** Write a function to find the Minimum Spanning Treeof a graph using Prim's algorithm.
37	**Fibonacci Sequence:** Write a function to compute the nth Fibonacci number using dynamic programming.
38	**Climbing Stairs:** Write a function to determine how many distinct ways there are to climb a staircase with n steps if you can climb either 1 or 2 steps at a time.
39	**Min Cost Climbing Stairs:** Write a function to determine the minimum cost to reach thetop of a staircase given a list of costs associated with each step.
40	**House Robber:** Write a function to determine the maximum amount of money you can rob from a row of houses without robbing two adjacent houses.
41	**Maximum Subarray Sum (Kadane's Algorithm):** Write a function to find the contiguous subarray with the maximum sum.
42	**Activity Selection:** Given a set of activities with start and end times, select the maximum number of activities that do not overlap.
43	**Fractional Knapsack Problem:** Given weights and values of items and the maximum capacity of a knapsack, determine the maximum value that can be obtained by including fractions of items.
44	**Huffman Coding:** Given a set of characters and their frequencies, construct the HuffmanTree to encode the characters.
45	**Job Sequencing Problem:** Given a set of jobs, each with a deadline and profit, maximize the total profit by scheduling the jobs to be done before their deadlines.
46	**Minimum Number of Coins:** Given different denominations of coins and an amount, findthe minimum number of coins needed to make up that amount.
47	**N-Queens Problem:** Place N queens on an N×N chessboard so that no two queens threaten each other.

48	**Permutations:** Generate all possible permutations of a given list of numbers or characters.
49	**Subsets:** Generate all possible subsets of a given set of numbers.

8. COURSE OUTCOMES(COs)

After the completion of the course, the student will be able to:

CO No.	Course Outcomes
R1UC503.1	Implement Stack, Queue, and Binary Tree data structures using Array and Linked Lists.
R1UC503.2	Analyze the time and space complexity of algorithms related to algorithm paradigms of divide and conquer, dynamic programming, greedy algorithms, and back tracking.
R1UC503.3	Develop programs for computational problems like shortest path, all pairs shortest path, N queens' problem, minimum spanning tree, longest common subsequence, 0/1 knapsack, choosing appropriate algorithmic techniques out of dynamic programming, greedy algorithms, and backtracking.
R1UC503.4	Develop simple software applications using basic and advanced data structures with dynamic programming, greedy algorithms, and backtracking.

9. TAXONOMY LEVEL OF THE COURSE OUTCOMES

Mapping of CO with Bloom's Level.

CO No.	Remember **KL1**	Understand **KL 2**	Apply **KL 3**	Analyse **KL 4**	Evaluate **KL 2**	Create **KL 6**
R1UC 503.1			√			
R1UC 503.2			√			
R1UC 503.3			√			
R1UC 503.4			√	√	√	√

10. COURSE ARTICULATIONMATRIX

TheCoursearticulationmatrixindicatesthecorrelationbetweenCourseOutcomesandProgram Outcomesandtheirexpectedstrengthofmappinginthreelevels(low,medium,andhigh).

COs#/ POs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
R1UC 503.1	2				2									
R1UC 503.2	2	2			2									
R1UC 503.3	2	2	1		2									
R1UC 503.4	2	2	2		2				1	1	1	1	2	

Note: 1-Low, 2-Medium, 3-High

11. TYPICAL EXAMPLE OF COURSES, CREDIT HOURS AND TEACHING HOURS

Type of Course	Credits Hours					Hours of engagement/ Week					15 weeks/ semester	Remarks
	Theory	Tutorial	Practical	Self-study	Total	Theory	Tutorial	Practical	Self-study	Total	Total no. of classes	
Integrated Course	3	0	1	0	5	3	0	2	0	5	75	45 classes for theory & 30 hours of lab sessions

SL-No	Topic for Delivery	Theory/ Practical Plan	Skill	Competency
1	Definition and significance of algorithmic complexity Time and space complexity, Big O notation, Omega, and Theta notations.	Lecture	The students will develop a strong foundation in algorithm analysis, complexity evaluation, and practical coding skills. These skills are essential for solving real-world problems efficiently.	CO1, CO2
2	Analyzing basic algorithms, solve problems on basic time complexity analysis.	Lecture		
3	Static vs. dynamic arrays. Find the Maximum and Minimum Elements in an Array.	Lecture		
4	Reverse an Array, Find the Kth Smallest/Largest Element in an Array	Lecture		
5	Write a function to find the maximum and minimum elements in an array. Write a function to find the Kth smallest or largest element in an array.	Practical		

6	Given an array containing only 0s, 1s, and 2s, sort the array in linear time. Write a function to move all zeroes in an array to the end while maintaining the relative order of other elements.	Practical		
7	Implementation of Singly Linked List.	Lecture	The students will gain a comprehensive understanding of linked lists, their implementations, and their operations. This knowledge is crucial for tackling more advanced data structures and algorithms in computer science.	CO1, CO2
8	Implementation of doubly linked list.	Lecture		
9	Implementation of circular linked list.	Lecture		
10	Operations on Linked List. Insertion, Deletion, Traversal.	Lecture		
11	Write a function to reverse a singly linked list. Write a function to detect if a cycle exists in a linked list.	Practical		
12	Write a function to find the middle element of a linked list. Write a function to merge two sorted linked lists into one sorted linked list.	Practical		
13	Implementation of Stack using linked list.	Lecture	The students will gain a comprehensive understanding of stacks, recursion, and their applications. These skills are crucial for tackling more advanced data structures and	CO1, CO2
14	Application of stack, infix, Prefix and Postfix Expressions.	Lecture		
15	Infix to postfix expression, Evaluation of postfix expression	Lecture		
16	Principles of recursion, Tail recursion, problem solving using recursion.	Lecture		
17	Write a function to implement a stack using an	Practical		
	array or list with basic operations: push, pop, peek, and isEmpty. Write a function to check if a string containing parentheses is balanced.		algorithms in computer science	

18	Write a function to evaluate a given postfix expression. Write a function to find the next greater element for each element in an array.	Practical		
19	Fibonacci numbers, a^n (a raised to the power n), reverse of a string.	Lecture	The students will gain understanding of queue data structure, hashing, recursion, and their applications.	CO2, CO3
20	Implementation and operations on Queue using linked list.	Lecture		
21	Priority queue (heap).	Lecture		
22	Concept of Hashing.	Lecture		
23	Write a function to implement a queue using a linked list with basic operations: enqueue, dequeue, front, and isEmpty. Write a function to implement a circular queue with basic operations: enqueue, dequeue, front, rear, and isEmpty.	Practical		
24	Write a function to generate binary numbers from 1 to N using a queue. Write a function to implement a queue using two stacks. (vice-versa)	Practical		
25	Collision resolution Techniques used in Hashing.	Lecture	The students will gain understanding of tree data structures, algorithms, and their applications.	CO1, CO3
26	Linked List Representation of trees, Binary Search Tree.	Lecture		
27	Tree Traversal algorithms: In-order, Pre-order, and Post-order.	Lecture		
28	Constructing Binary Tree from given Tree Traversal,	Lecture		

29	Write a class to implement a basic binary tree with insert, delete, and traversal operations. Write a functions to perform in-order, pre-order, and post-order traversal of a binary tree.	Practical		
30	Write a function to find the diameter of a binary tree. Write a function to check if a binary tree is height balanced. Write a function to find the height of a binary tree.	Practical		
31	Operation of Insertion, Deletion, Searching Modification of data in Binary Search	Lecture	The students will gain a comprehensive understanding of binary search trees, graph theory, and their applications. These skills are crucial for tackling more advanced data structures and algorithms in computer science.	CO1, CO3
32	Graph representations: Adjacency Matrices, Adjacency List, Graph Traversal: Depth First Search	Lecture		
33	Breadth First Search.	Lecture		
34	Detect Cycle in an Undirected Graph, Connected components in an Undirected Graph.	Lecture		
35	Write a function to find the lowest common ancestor of two nodes in a binary tree. Write a class to implement a basic graph using an adjacency list with methods to add vertices and edges. Write a function to perform DFS on a graph from a given start vertex.	Practical		
36	Write a function to detect if there is a cycle in an undirected graph. Write a function to perform BFS on a graph from a given start vertex.	Practical		

	Write a function to find the number of connected components in an undirected graph.			
37	Minimum spanning trees, Prims and Kruskal algorithm.	Lecture	The students will gain a comprehensive understanding of advanced algorithms, dynamic programming techniques, and their applications.	CO2, CO3
38	Shortest Path algorithm: Warshal Algorithm.	Lecture		
39	Elements of dynamic programming, longest common subsequence	Lecture		
40	Coin change problem.	Lecture		
41	Write a function to find the Minimum Spanning Tree of a graph using Kruskal's algorithm. Write a function to find the Minimum Spanning Tree of a graph using Prim's algorithm.	Practical		
42	Write a function to compute the nth Fibonacci number using dynamic programming. Write a function to determine how many distinct ways there are to climb a staircase with n steps if you can climb either 1 or 2 steps at a time.	Practical		
43	Determine the minimum cost to reach the top of a staircase given a list of costs associated with each step.	Lecture	The students will gain a comprehensive understanding of dynamic programming, greedy algorithms, and their applications.	CO1, CO3
44	To find the contiguous subarray with the maximum sum.	Lecture		
45	Elements of the greedy strategy, Fractional knapsack,	Lecture		
46	Activity Selection, Huffman coding.	Lecture		
47	Given weights and values of items and the maximum capacity of a knapsack, determine the maximum value that can be obtained by including fractions of items.	Practical		

	Given a set of characters and their frequencies, construct the Huffman Tree to encode the characters.			
48	Write a function to determine the maximum amount of money you can rob from a row of houses without robbing two adjacent houses. Given n ropes with different lengths, find the minimum cost to connect all ropes into one rope.	Practical		
49	Job sequencing Problem	Lecture	The students will gain a comprehensive understanding of optimization problems, backtracking algorithms, and their applications. These skills are crucial for tackling complex problems in computer science and software engineering, particularly in areas requiring combinatorial and recursive problem-solving.	CO3
50	Given different denominations of coins and an amount, find the minimum number of coins needed to make up that amount.	Lecture		
51	Back Tracking: Sum of subset	Lecture		
52	N queens' problem	Lecture		
53	Place N queens on an N×N chessboard so that no two queens threaten each other Generate all possible permutations of a given list of numbers or characters.	Practical		
54	Generate all possible subsets of a given set of numbers Rat in a maze: Given a maze, find a path from the top-left corner (0, 0) to the bottom-right corner (n-1, n-1). The rat can move in four directions: left, right, up, and down.	Practical		
55	Introduction to Probabilistic Algorithms, Advantages and Trade-offs of Probabilistic Data Structures.	Lecture	The students will be well-equipped to handle problems that require efficient space and time management	CO3, CO4
56	Applications and Use Cases, Structure and Function of Bloom Filters.	Lecture		

57	Hash Functions and Their Role, False Positives and Space Efficiency.	Lecture	through probabilistic methods, and they will understand how to apply these techniques in real-world scenarios.	
58	Variants of Bloom Filters (e.g., Counting Bloom Filter).	Lecture		
59	Revision	Lecture		
60		Lecture		
61		Lecture		
62		Lecture		

12. BIBLIOGRAPHY

- **Text Book:** Introduction to Algorithms, by Corman
- **Reference Books**
  - Data Structures and Algorithms in Java, Roberto Tamassia and Michael T Goodrich, John Wiley
  - R. Kruse etal, "Data Structures and Program Design in C", Pearson Education
- **Webliography**
  - https://www.geeksforgeeks.org/advanced-data-structures/
  - https://github.com/topics/advanced-data-structures
- **SWAYAM/NPTEL/MOOCs Certification**
  - https://www.coursera.org/specializations/data-structures-algorithms.
  - https://www.codespaces.com/best-data-structures-and-algorithms-courses-classes.html#3-data-structures-and-algorithms-nanodegree-certification-udacity

13. COURSE ASSESSMENT

Assessment forms an integral part of curriculum design. A learning-teaching system can only be effective if the student's learning is measured at various stages which means while the student processes learning (Assessment for Learning) a given content and after completely learning a defined content (Assessment of Learning). Assessment for learning is referred to as formative assessment, that is, an assessment designed to inform instruction.

The ability to use and apply the knowledge in different ways may not be the focus of the assessment. With regard to designing assessments, the faculty members must be willing to put in the time required to create a valid, reliable assessment, that ideally would allow students to demonstrate their understanding of the information while remaining. The following are the five main areas that assessment reporting should cover.

1. **Learning Outcomes**: At the completion of a program, students are expected to know their knowledge, skills, and attitude. Depending on whether it is a UG or PG program, the level of sophistication may be different. There should be no strict rule on the number of outcomes to be achieved, but the list should be reasonable, and well-organized.

2. **Assessable Outcomes**: After a given learning activity, the statements should specify what students can do to demonstrate. Criteria for demonstration are usually addressed in rubrics and there should be specific examples of work that doesn't meet expectations, meets expectations, and exceeds expectations. One of the main challenges is faculty communication whether all faculty agreed on explicit criteria for assessing each outcome. This can be a difficult accomplishment when multiple sections of a course are taught or different faculty members. Hence there is a need for common understanding among the faculty on what is assessed and how it is assessed.
3. **Assessment Alignment**: This design of an assessment is sometimes in the form of a curriculum map, which can be created in something as easy as an Excel spreadsheet. Courses should be examined to see which program outcomes they support, and if the outcome is assessed within the course. After completion, program outcomes should be mapped to multiple courses within the program.
4. **Assessment Planning**: Faculty members need to have a specific plan in place for assessing each outcome. Outcomes don't need to be assessed every year, but faculty should plan to review the assessment data over a reasonable period of time and develop a course of action if the outcome is not being met.
5. **Student Experience**: Students in a program should be fully aware of the expectations of the program. The program outcomes are aligned on the syllabus so that students are aware of what course outcomes they are required to meet, and how the program outcomes are supported. Assessment documents should clearly communicate what is being done with the data results and how it is contributing to the improvement of the program and curriculum.

Designing quality assessment tools or tasks involves multiple considerations if it is to be fit for purpose. The set of assessments in a course should be planned to provide students with the opportunity to learn as they engage with formative tasks as well as the opportunity to demonstrate their learning through summative tasks. Encouraging the student through the use of realistic, authentic experiences is an exciting challenge for the course faculty team, who are responsible for the review and quality enhancements to assessment practices.

14. FORMATIVE AND SUMMATIVE ASSESSMENT

Assessment Pattern for Integrated Course:

Type of course	CIE			Total Marks		Final Marks CIE*0.5+SEE*0.5
	Lab@ (Work + Record)	MTE	Lab Exam	CIE	SEE	
Comprehensive	25	50	25	100	100	100

@Lab Work-15 marks + Lab Record-10 marks

15. PASSING STANDARDS

Passing Criteria for Different Course Types Effective from AY 2022-23 Onwards

S.No.	Course Type	Passing Criterion
1.	**Comprehensive Course (C)**	A student shall secure a minimum of **30% of the maximum marks** in the semester-end examination (SEE/ETE) and **40% of aggregate marks** in the course Continuous internal examination (CIE) and SEE/ETE marks i.e., minimum Passing Grade in a course is "P".

Note: Students unable to meet the overall passing criteria as mentioned shall be eligible for the following options to clear the course:

- **Appear in the Back Paper Examinations and have to meet the criteria to score 40% in marks overall**
- **Appear in summer examinations (Internal +External) to meet the criteria as mentioned.**

16. PROBLEM-BASED LEARNING/CASE STUDIES/CLINICS

SNo	Problem	KL
1	Write a program in Java or Python and find time complexity for 2–Sum Problem: Finding two numbers in an array that add up to a given target value.	K3
2	Write a program in Java or Python and find time complexity for Longest Common Subsequence Problem: Finding longest subsequence which is common in all given input sequences.	K3
3	Write a program in Java or Python and find time complexity for Maximum Subarray Problem: Finding a contiguous subarray with the largest sum, within a given one-dimensional array A[1...n] of numbers.	K3
4	Write a program in Java or Python and find time complexity for Coin Change Problem: Finding the number of ways to make sum by using different denominations from an integer array of coins[ ] of size N representing different types of denominations and an integer sum.	K3
5	Write a program in Java or Python and find time complexity for 0–1 Knapsack Problem: Restrict the number of copies of each kind of item to zero or one.	K3
6	Write a program in Java or Python and find time complexity for Subset Sum Problem: Checking if there is a subset of the given set whose sum is equal to the given sum.	K3
7	Write a program in Java or Python and find time complexity for Longest Palindromic Subsequence Problem: Finding a maximum-length subsequence of a given string that is also a Palindrome.	K3
8	Write a program in Java or Python and find time complexity for Matrix Chain Multiplication Problem: Finding the most efficient way to multiply these matrices together such that the total number of element multiplications is minimum.	K3

9	Write a program in Java or Python and find time complexity for Longest Common Substring Problem: A set of strings can be found by building a generalized suffix tree for the strings, and then finding the deepest internal nodes which have leaf nodes from all the strings in the subtree below it.	K3
10	Write a program in Java or Python and find time complexity for Rod Cutting Problem: A rod is given of length n. Another table is also provided, which contains different size and price for each size. Determine the maximum price by cutting the rod and selling them in the market.	K3
11	Write a program in Java or Python and find time complexity for Word Break Problem: You will be given a string, say "s", and a dictionary of strings say "wordDict". You have to return true if s can be segmented into a space-separated sequence of one or more dictionary words.	K3
12	Write a program in Java or Python and find time complexity for Edit Distance Problem: Quantifying how dissimilar two strings (e.g., words) are to one another, that is measured by counting the minimum number of operations required to transform one string into the other.	K3
13	Write a program in Java or Python and find time complexity for Chess Knight Problem: A knight starting at any square of the board and moving to the remaining 63 squares without ever jumping to the same square more than once.	K3
14	Write a program in Java or Python and find time complexity for Partition Problem: A given set can be partitioned in such a way, that sum of each subset is equal.	K3
15	Write a program in Java or Python and find time complexity for 3–Partition Problem: Deciding whether a given multiset of integers can be partitioned into triplets that all have the same sum.	K3
16	Write a program in Java or Python and find time complexity for Snake and Ladder Problem: Write a function that returns the minimum number of jumps to take top or destination position. You can assume the dice you throw results in always favor of you means you can control the dice.	K3
17	Write a program in Java or Python and find time complexity for Largest Consecutive Subarray Problem: Finding out the largest sum of the consecutive numbers of the array.	K3
18	Write a program in Java or Python and find time complexity for Dutch National Flag Problem: The flag of the Netherlands consists of three colors: white, red,	K3
	and blue. The task is to randomly arrange balls of white, red, and blue such that balls of the same color are placed together.	
19	Write a program in Java or Python and find time complexity for Knight's Tour Problem: a puzzle where a chess knight is placed on an empty chess board and the goal is to move the knight to every square on the board exactly once without re-visiting any squares.	K3
20	Write a program in Java or Python and find time complexity for Maximum Sum Submatrix Problem: A 2D array arr[][] of dimension N*M is given, the task is to find the maximum sum sub-matrix from the matrix arr[][].	K3
21	Write a program in Java or Python and find time complexity for Longest Palindromic Substring Problem: Finding a maximum-length contiguous substring of a given string that is also a palindrome.	K3

22	Write a program in Java or Python and find time complexity for Job Sequencing Problem: You have a single processor operating system and a set of jobs that have to be completed with given deadline constraints.	K3
23	Write a program in Java or Python and find time complexity for N–Queens Problem: Placing N chess queens on an N×N chessboard so that no two queens attack each other.	K3
24	Write a program in Java or Python and find time complexity for Maximum Product Subarray Problem: Find the contiguous subarray within the array which has the largest product of its elements. You have to report this maximum product.	K3
25	Write a program in Java or Python and find time complexity for Longest Repeated Subsequence Problem: Find the length of the longest repeating subsequence in a given string such that the two subsequences don't have the same original string character at the same position.	K3
26	Write a program in Java or Python and find time complexity for 3–Sum Problem: Given an array and a value, find if there is a triplet in array whose sum is equal to the given value. If there is such a triplet present in array, then print the triplet and return true. Else return false.	K3
27	Write a program in Java or Python and find time complexity for Shortest Common Super Sequence Problem: Given two strings X and Y of lengths m and n respectively, find the length of the smallest string which has both, X and Y as its sub-sequences.	K3
28	Write a program in Java or Python and find time complexity for Longest Alternating Subarray Problem: Given an array containing positive and negative elements, find a subarray with alternating positive and negative elements, and in which the subarray is as long as possible.	K3
29	Write a program in Java or Python and find time complexity for 4–Sum Problem: Given an array nums of n integers and an integer target, are there elements a, b, c, and d in nums such that a + b + c + d = target we need to find all unique quadruplets in the array which gives the sum of target.	K3
30	Write a program in Java or Python and find time complexity for K–Partition Problem: Partitioning an array of positive integers into k disjoint subsets that all have an equal sum, and they completely cover the set.	K3
31	Write a program in Java or Python and find time complexity for Minimum Sum	K3
	Partition Problem: Given a set of positive integers S, partition set S into two subsets, S1 and S2, such that the difference between the sum of elements in S1 and S2 is minimized. The solution should return the minimum absolute difference between the sum of elements of two partitions.	
32	Write a program in Java or Python and find time complexity for Wildcard Pattern Matching Problem: We have a string and a pattern then we have to compare the string with a pattern that whether the pattern matches with a string or not	K3
33	Write a program in Java or Python and find time complexity for Maximum Overlapping Intervals Problem: Print the maximum number of overlap among these intervals at any time.	K3
34	Write a program in Java or Python and find time complexity for Graph Coloring Problem: Assigning colors to the vertices such that no two adjacent vertexes have the same color.	K3

35	Write a program in Java or Python and find time complexity for Longest Increasing Subsequence Problem: Given an array **arr[]** of size **N**, the task is to find the length of the Longest Increasing Subsequence (LIS).	K3
36	Write a program in Java or Python and find time complexity for Pots of Gold Game Problem: Two players X and Y are playing a game in which there are pots of gold arranged in a line, each containing some gold coins. They get alternating turns in which the player can pick a pot from one of the ends of the line. The winner is the player who has a higher number of coins at the end. The objective is to maximize the number of coins collected by X, assuming Y also plays optimally. Return the maximum coins X could get while playing the game. Initially, X starts the game.	K3
37	Write a program in Java or Python and find time complexity for Activity Selection Problem: Selection of non-conflicting activities that needs to be executed by a single person or machine in a given time frame.	K3
38	Write a program in Java or Python and find time complexity for Longest Alternating Subsequence Problem:One wants to find a subsequence of a given sequence in which the elements are in alternating order, and in which the sequence is as long as possible.	K3
39	Write a program in Java or Python and find time complexity for Longest Consecutive Subsequence Problem: First sort the array and find the longest subarray with consecutive elements. After sorting the array and removing the multiple occurrences of elements, run a loop and keep a count and max (both initially zero).	K3
40	Write a program in Java or Python and find time complexity for Weighted Interval Scheduling Problem: A value is assigned to each executed task and the goal is to maximize the total value. The solution need not be unique.	K3
41	Write a program in Java or Python and find time complexity for Longest Bitonic Subarray Problem: Find a subarray of a given sequence in which the subarray's elements are first sorted in increasing order, then in decreasing order, and the subarray is as long as possible.	K3
42	Write a program in Java or Python and find time complexity for Water Jugs	K3
	Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?	
43	Write a program in Java or Python and find time complexity for Hat Check Problem: Given a positive number n, find the total number of ways in which n hats can be returned to n people such that no hat makes it back to its owner.	K3
44	Write a program in Java or Python and find time complexity for Merging Overlapping Intervals: Start from the first interval and compare it with all other intervals for overlapping.	K3
45	Write a program in Java or Python and find time complexity for Longest Common Prefix (LCP) Problem: An array of strings is the common prefix between 2 most dissimilar strings.	K3

17. SELF-LEARNING THROUGH MOOCs (Cognitive Skills): Certifications

1. **"Algorithms Specialization" by Stanford University on Coursera:** This specialization covers multiple courses on algorithms and data structures, including "Divide and Conquer, Sorting, and Searching" and "Graph Search, Shortest Paths, and Data Structures."
2. **"Data Structures and Algorithms" by University of California San Diego & National Research University Higher School of Economics on Coursera:** This course covers essential data structures and algorithms, and the programming assignments are often in C++.
3. **"Data Structures and Algorithms - The Complete Masterclass" on Udemy:** This course covers a wide range of data structures and algorithms topics using C++ and includes coding exercises.
4. **"Mastering Data Structures & Algorithms using C and C++" on Udemy:** Another comprehensive course that covers data structures and algorithms with a focus on C and C++ programming languages.
5. **"Data Structures and Algorithms in C++" by Coding Blocks on YouTube:** This is a free YouTube series that covers data structures and algorithms in C++.

Course Lead **Programme Chair** **Dean (SCSE)**