Government of Karnataka

DEPARTMENT OF COLLEGIATE AND TECHNICAL EDUCATION

Programme	Computer Science and Engineering	**Semester**	IV
Course Code	20CS41P	**Type of Course**	Programme Core
Course Name	Data Structures with Python	**Contact Hours**	8 hours/week 104 hours/semester
Teaching Scheme	L:T:P :: 3:1:4	**Credits**	6
CIE Marks	60	**SEE Marks**	40

1.Rationale

Data structures are the techniques organizing data and of designing the algorithms for real-life projects. Knowledge of data structures is essential for software design and development. Learning data structures with Python offer flexibility and ease of programming with many built in data structures and libraries.

2. Course Outcomes: At the end of the Course, the student will be able to:

CO-01	Explain data structures types, list their applications.
CO-02	Apply the right Algorithm design strategies to solve a given problem.
CO-03	Choose the right data structure to develop solution to a given computing problem.
CO-04	Analyse space and time complexities of the algorithm used and plot a graph.

3. Course Content

Week	CO	PO	Lecture (Knowledge Criteria)	Tutorial (Activity Criteria)	Practice (Performance Criteria)
			3 hours/week	1 hour/week	4 hours/week(2 hours/batch twice in a week)
1	01	1, 2, 3	Introduction to Data Structures, operations, classification, Characteristics. Primitive types – primitive data structures, python examples. Non primitive types - Non primitive data structures, python examples. Linear and nonlinear data structures – with python examples. Introduction, Abstractions, Abstract Data Types, An Example of Abstract Data Type (Student, Date, Employee), Defining the ADT, Using the ADT, Implementing the ADT.	Refer Table 1	1. Python program to Use and demonstrate basic data structures. 2. Implement an ADT with all its operations.
2	01, 02, 04	1, 2, 3, 4, 7	Algorithm Analysis – Space Complexity, Time Complexity. Run time analysis. Asymptomatic notations, Big-O Notation, Omega Notation, Theta Notation.		1. Implement an ADT and Compute space and time complexities. 2. Implement above solution using array and Compute space and time complexities and compare two solutions.

3	01, 02, 04	1, 2, 3, 4, 7	Algorithm design strategies: Brute force – Bubble sort, Selection Sort, Linear Search. Decrease and conquer - Insertion Sort.		1. Implement Linear Search compute space and time complexities, plot graph using asymptomatic notations. 2. Implement Bubble, Selection, insertion sorting algorithms compute space and time complexities, plot graph using asymptomatic notations.
4	01, 02, 04	1, 2, 3, 4, 7	Divide and conquer - Merge Sort, Quick Sort, Binary search. Dynamic programming - Fibonacci sequence Backtracking – Concepts only (Implementation examples with recursion in week 9). Greedy – Concepts only.		1. Implement Binary Search using recursion Compute space and time complexities, plot graph using asymptomatic notations and compare two. 2. Implement Merge and quick sorting algorithms compute space and time complexities, plot graph using asymptomatic notations and compare all solutions. 3. Implement Fibonacci sequence with dynamic programing.
5	01, 02, 03, 04	1, 2, 3, 4,	Linear(arrays) vs nonlinear (pointer) structures – Run time and space requirements, when to use what? Introduction to linked list, Examples: Image viewer, music player list etc. (to be used to explain concept of list), applications.		1. Implement Singly linked list (Traversing the Nodes, searching for a Node, Prepending Nodes, Removing Nodes)
6	01, 02, 03, 04	1, 2, 3, 4,	The Singly Linked List- Creating Nodes, Traversing the Nodes, searching for a Node, Prepending Nodes, Removing Nodes. Linked List Iterators.		1. Implement linked list Iterators.
7	01, 02, 03, 04	1, 2, 3, 4,	The Doubly Linked List, Examples: Image viewer, music player list etc. (to be used to explain concept of list). DLL node, List Operations – Create, appending nodes, delete, search.		1. Implement DLL. 2. Implement CDLL

			The Circular Linked List-Organization, List Operations – Appending nodes, delete, iterating circular list.		
8	01, 02, 03, 04	1, 2, 3, 4	Last In First Out (Stack) Data structures – Example: Reversing a word, evaluating an expression, message box etc. (to be used to explain concept of LIFO). The Stack implementation – push, pop, display. Stack Applications- Balanced Delimiters, Evaluating Postfix Expressions.		1. Implement Stack Data Structure. 2. Implement bracket matching using stack.
9	01, 02, 03, 04	1, 2, 3, 4,	Recursion. Properties of Recursion. Recursive functions: Factorials, Recursive Call stack, The Fibonacci Sequence. How Recursion Works- The Run Time Stack. Recursive Applications- Recursive Binary Search, Towers of Hanoi.		1. Program to demonstrate recursive operations (factorial/ Fibonacci) 2. Implement solution for Towers of Hanoi.
10	01, 02, 03, 04	1, 2, 3, 4,	The First In First Out (Queue) Data structure – Example: Media player list, keyboard buffer queue, printer queue etc. (to be used to explain concept of FIFO). Implementing the Queue and its operations using Python List. Priority Queues, Implementation.		1. Implement Queue. 2. Implement priority queue
11	01, 02, 03, 04	1, 2, 3, 4,	The Tree data structure – Example: File explorer/Folder structure, Domain name server. Tree Terminologies, Tree node representation. Binary trees, Binary search trees, Properties, Implementation of tree operations – insertion, deletion, search, Tree traversals (in order, pre order and post order).		1. Implement Binary search tree and its operations using list.
12	01, 02, 04	1, 2, 3, 4,	Depth-first traversal Breadth-first traversal Tree applications: Expression evaluation.		1. Implementations of BFS. 2. Implementation of DFS.
13	01, 03, 04	1, 2, 3, 4,	Introduction to Hashing. Hashing - Perfect hashing functions. Hash table Hash Functions, Operations, Hash collision, Application.		1. Implement Hash functions.
Total in hours			**39**	**13**	**52**

***PO = Program outcome as listed and defined in year 1 curriculum**

Table 1: Suggestive activities for tutorials (the list is only shared as an example and not inclusive of all possible activities for that course. Student and faculty are encouraged to choose activities that are relevant to the topic and the availability of such resources at their institution)

Sl. No	Activity
1	Design a Data structure for handling Student Records- Designing a Solution, Implementation (Using Basic DS).

2	Design a Data structure for handling Student Records- Designing a Solution, Implementation (Using ADT).
3	Optimize your solution (Bubble sort, selection sort and Insertion sort)
4	Implement Radix sort.
5	Prepare report on nonlinear data structures.
6	Design and implement sparse matrix representation using linked list.
7	Design and implement simple application that require DLL data structure.
8	Implement and demonstrate evaluating postfix expression.
9	Presentation on run time stack.
10	Design and implement priority queue data structure.
11	Prepare a Report on balanced trees.
12	Implement expression evaluation tree.
13	Prepare a report on hashing and analyze time complexity.

4. CIE and SEE Assessment Methodologies

Sl. No	**Assessment**	**Test Week**	**Duration** In minutes	**Max marks**	**Conversion**
1.	CIE-1 Written Test	5	80	30	Average of three tests 30
2.	CIE-2Written Test	9	80	30	
3	CIE-3Written Test	13	80	30	
4.	CIE-4 Skill Test-Practice	6	180	100	Average of two skill tests reduced to 20
5	CIE-5 Skill Test-Practice	12	180	100	
6	CIE-6 Portfolio continuous evaluation of Activity through Rubrics	1-13		10	10
				Total CIE Marks	60
	Semester End Examination (Practice)		180	100	40
				Total Marks	**100**

5. Format for CIE written Test

Course Name	**Data Structures with Python**	Test	I/II/III	Sem	III/IV
Course Code	**20CS41P**	Duration	80 Min	Marks	30

Note: Answer any one full question from each section. Each full question carries 10 marks.

Section	Assessment Questions	Cognitive Levels	Course Outcome	Marks
I	1			
	2			
II	3			
	4			
III	5			
	6			

Note for the Course coordinator: Each question may have one, two or three subdivisions. Optional questions in each section carry the same weightage of marks, Cognitive level and course outcomes.

6. Rubrics for Assessment of Activity (Qualitative Assessment)

	Dimension	Beginner	Intermediate	Good	Advanced	Expert	

Sl. No.		2	4	6	8	10	Students Score
1		Descriptor	Descriptor	Descriptor	Descriptor	Descriptor	8
2		Descriptor	Descriptor	Descriptor	Descriptor	Descriptor	6
3		Descriptor	Descriptor	Descriptor	Descriptor	Descriptor	2
4		Descriptor	Descriptor	Descriptor	Descriptor	Descriptor	2
	Average Marks=(8+6+2+2)/4=4.5						**5**

Note: Dimension and Descriptor shall be defined by the respective course coordinator as per the activities

7. Reference:

Sl. No.	Description
1	Data Structures and Algorithms using Python by Rance D. Necaise
2	Python Data Structures and Algorithms by Benjamin Baka
3	www.geeksforgeeks.com

8. CIE Skill Test and SEE Scheme of Evaluation

SL. No.	Particulars/Dimension	Marks
1	Select appropriate Algorithm design strategy to solve the given problem statement, and design Algorithmic solution.	25
2	Develop the solution choosing appropriate data structure, test and debug the solution.	30
3	Analyse time and space complexity and plot the graph.	15
4	Demonstrate the solution and its operation, justify your selection of Algorithm /Data structure using above graphs. In the event of, a student fails to get the desired result (with no syntactical and least semantic errors), the examiner shall use viva voce to assess the student's understanding of different data structures and algorithm design strategies.	20
5	Portfolio evaluation based on aggregate of all practice sessions	10
	Total Marks	**100**

9. Equipment/software list with Specification for a batch of 20 students

Sl. No.	Particulars	Specification	Quantity
1	Computers		20
2	Python 3.6		20
3	Editor such as iPython, Jupyter, spider, PyCharm		20