COURSE CONTENT

DATA STRUCTURES LAB USING PYTHON

III Semester: CE / CSD / CSE / CSM / ECE / EEE / ME

Course Code	Category	Hours / Week			Credits	Maximum Marks		
		L	T	P	C	CIA	SEE	Total
2430575	Foundation	0	0	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 30				Total Classes: 30		

Prerequisites: "Essentials of Problem Solving using Python".

Course Overview:

Data structures are the fundamental building blocks of computer programming. They define how data is organized, stored, and manipulated within a program. Understanding data structures is very important for developing efficient and effective algorithms. In this Course, student will explore the most commonly used data structures, including linked **lists, stacks, queues, trees, and Hashing.**

Course Objectives:

The students will try to learn

- Various linear and non-linear data structures.
- How to perform operations on data structures.
- Priority Queues and Heaps
- Various searching and sorting techniques.
- Different hashing techniques

Course Outcomes: After Completion of the Course, Students should be able to

- Implement linear data structures such as singly, doubly, and circular linked lists and apply them for solving real-world problems.
- Design stack and queue data structures using arrays and linked lists, and apply them in practical applications such as expression evaluation and scheduling.
- Construct and manipulate tree-based data structures including binary trees, binary search trees, AVL trees, B-trees, and heaps, and apply them in searching and priority queue applications.
- Apply appropriate searching and sorting algorithms, analyze their time and space complexities, and select suitable techniques for different problem scenarios.
- Design efficient hashing techniques using suitable hash functions and collision resolution strategies to optimize data retrieval and storage operations.

List of Experiments

1. Write a program that uses functions to perform the following operations on singly linked list.: i) Creation ii) Insertion iii) Deletion iv) Traversal

2. Write a program that uses functions to perform the following operations on doubly linked list.: i) Creation ii) Insertion iii) Deletion

3. Write a program that uses functions to perform the following operations on circular linked list: i) Creation ii) Insertion iii) Deletion

4. Write a program that implement stack operations using i) Arrays ii) Pointers

5. Write a c program to implement infix to postfix conversion using stack.

6. Write a c program to implement postfix evaluation.

7. Write a program that implement Queue operations using i) Arrays ii) Pointers

8. Write a program to implement the tree traversal methods using both recursive and non-recursive.

9. Write a program to implement tree operations on i) AVL Trees ii) B Trees iii) Heap

10. Write a program that implements the following sorting methods to sort a given list of integers in ascending order i) Bubble sort ii) Selection sort iii) Insertion sort

11. Write a program that implements the following sorting methods to sort a given list of integers in ascending order i) Merge sort ii) Quick sort iii) Heap Sort

12. Write a program that use both recursive and non-recursive functions to perform the following searching operations for a Key value in a given list of integers: i) Linear search ii) Binary search

13. Write a program to implement hashing**.**

CASE STUDY-1 Balanced Brackets

A bracket is considered to be any one of the following characters: (,), {, }, [, or].

Two brackets are considered to be a *matched pair* if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e.,),], or }) *of the exact same type*. There are three types of matched pairs of brackets: [], {}, and ().

A matching pair of brackets is *not balanced* if the set of brackets it encloses are not matched. For example, {[(])} is not balanced because the contents in between { and } are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, (, and the pair of parentheses encloses a single, unbalanced closing square bracket,].

By this logic, we say a sequence of brackets is *balanced* if the following conditions are met:

- It contains no unmatched brackets.
- The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets.

Given strings of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, return YES. Otherwise, return NO.

CASE STUDY-2 Minimum Average Waiting Time

Mr. Raju owns a pizza restaurant and he manages it in his own way. While in a normal restaurant, a customer is served by following the first-come, first-served rule, Raju simply minimizes the average waiting time of his customers. So he gets to decide who is served first, regardless of how sooner or later a person comes.

Different kinds of pizzas take different amounts of time to cook. Also, once he starts cooking a pizza, he cannot cook another pizza until the first pizza is completely cooked. Let's say we have three customers who come at time t=0, t=1, & t=2 respectively, and the time needed to cook their pizzas is 3, 9, & 6 respectively. If Raju applies first-

come, first-served rule, then the waiting time of three customers is 3, 11, & 16 respectively. The average waiting time in this case is (3 + 11 + 16) / 3 = 10. This is not an optimized solution. After serving the first customer at time t=3, Raju can choose to serve the third customer. In that case, the waiting time will be 3, 7, & 17 respectively. Hence the average waiting time is (3 + 7 + 17) / 3 = 9.
Help Raju achieve the minimum average waiting time. For the sake of simplicity, just find the integer part of the minimum average waiting time.

Note:

- The waiting time is calculated as the difference between the time a customer orders pizza (the time at which they enter the shop) and the time she is served.
- Cook does not know about the future orders.

TEXTBOOKS:

1. Fundamentals of data structures in C, E.Horowitz, S.Sahni and Susan Anderson Freed, 2nd Edition, Universities Press.

2. Data structures using C, A.S.Tanenbaum, Y. Langsam, and M.J. Augenstein, PHI/pearson education.

REFERENCES:

1. Data structures: A Pseudocode Approach with C, R.F.Gilberg And B.A.Forouzan, 2nd Edition, Cengage Learning.

2. Introduction to data structures in C, Ashok Kamthane, 1st Edition, PEARSON

ELECTRONIC RESOURCES:

1. https://www.geeksforgeeks.org/data-structures/
2. https://www.programiz.com/python-programming/data-structure
3. https://realpython.com/tutorials/data-structures/
4. https://runestone.academy/ns/books/published/pythonds/index.html
5. https://www.studytonight.com/python/data-structures-in-python
6. https://www.tutorialspoint.com/python_data_structure/index.htm
7. https://www.w3schools.com/python/python_lists.asp

MATERIALS ONLINE:

1. Lab manual
2. Open-ended experiments