VES Institute of Technology
Department of Artificial Intelligence and Data Science
NADPC 32 : Data Structures - Practical Questions

Note : Each question will a problem to be solved based on the Data Structures Theory

Write a C program to implement

1. Stack using Arrays and demonstrate the Overflow & Underflow conditions
2. Parenthesis Matching using Stacks
3. Conversion of Infix Expression to Postfix Expression using Stacks
4. Postfix Expression evaluation using Stacks
5. Linear Queue using Arrays and demonstrate the Overflow & Underflow conditions
6. Circular Queue using Arrays and demonstrate Overflow & Underflow conditions
7. Priority Queue using Arrays and demonstrate the status after Insertions and Deletions
8. Singly Linked List - Insert at Beg, Delete from End, Display
9. Singly Linked List - Insert at End, Delete at Beg, Display
10. Singly Linked List - Insert at Beg, Delete at specific node, Display
11. Stack using Singly Linked List
12. Linear Queue using Singly Linked List
13. Circular Linked List - Insert at Beg, Delete from End, Display
14. Circular Linked List - Insert at End, Delete at Beg, Display
15. Circular Linked List - Insert at Beg, Delete a specific node, Display

Write a Java program to implement

16. Binary Search Tree - Insert values and perform Inorder Traversal
17. Binary Search Tree - Insert values and perform Postorder Traversal
18. Binary Search Tree - Insert values and perform Preorder Traversal
19. Binary Search Tree - Insert values and delete the node
20. Expression Tree, given the Postfix expression as input and print the Infix Expression
21. Binary Search for the given set of values. Compute the comparison time required for Successful Search and Unsuccessful Search
22. Hashing Function - Division Method. Count the number of Collisions and solve the Collision using Linear Probing
23. Hashing Function - Mid Square Method. Count the number of Collisions and solve the Collision using Linear Probing
24. Hashing Function - Multiplication Method. Count the number of Collisions and solve the Collision using Linear Probing
25. Selection Sort and count the number of comparisons and swaps respectively.
26. Insertion Sort and count the number of comparisons and swaps respectively.
27. Merge Sort and count the number of comparisons and swaps respectively.
28. Quick Sort and count the number of comparisons and swaps respectively.
29. Heap Sort and count the number of comparisons and swaps respectively.
30. Bucket Sort and count the number of comparisons and swaps respectively.