

Practical List

1. Conversion of Infix expression to postfix expression using stack.
2. Conversion of Infix expression to prefix expression using stack.
3. Write a program to maintain multiple queues in a single array.
4. Create a data structure **kQueues** that represents 'k' queues. Implementation of kQueues should use only one array, i.e., k queues should use the same array for storing elements. Following functions must be supported by kQueues.
 - a. enqueue(int x, int qn): adds x to queue number 'qn' where qn is from 0 to k-1.
 - b. dequeue(int qn): deletes an element from queue number 'qn' where qn is from 0 to k-1.
 - c. displayq(int q): displays all the elements of the q specified.
 - d. displayAll():displays the contents of all the queues.
5. Evaluation of Postfix expression using Stack
6. Evaluation of Prefix expression using Stack
7. Write a program to implement Queue using a circular array. The following operations need to be supported:
 - a. enqueue
 - b. dequeue
 - c. isEmpty
 - d. isFull
 - e. front
 - f. rear
8. Given a singly linked list, determine if it's a palindrome. Return 1 or 0 denoting if it's a palindrome or not, respectively. **For example**
 - a. List $1 \rightarrow 2 \rightarrow 1$ is a palindrome.
 - b. List $1 \rightarrow 2 \rightarrow 3$ is not a palindrome.
9. Implement univariate Polynomial addition using linked lists. A univariate polynomial should be represented in the decreasing order of its coefficient.

10. Implement univariate Polynomial Multiplication using linked lists. A univariate polynomial should be represented in the decreasing order of its coefficient.
11. Implement the following operations on doubly linked lists:
- Insert (after position)
 - sort the list
 - display
12. Implement the following operations on doubly linked lists:
- Insert (end)
 - concatenate two lists
 - display
13. Given a sorted linked list, delete all duplicates such that each element appears only once.
- For example:**
- Given $1 \rightarrow 1 \rightarrow 2$, return $1 \rightarrow 2$
 - Given $1 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 3$, return $1 \rightarrow 2 \rightarrow 3$
14. Given a linked list and a value x , partition it such that all nodes less than x come before nodes greater than or equal to x . You should preserve the original relative order of the nodes in each of the two partitions.
- For example:**
- Input: $1 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 2$ and $x = 3$
 - Output: $1 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4$
15. Write a program to construct a binary search tree and traverse it with all methods that uses recursion.
16. Write a program to represent the given graph using adjacency matrix and implement Breadth-First Search Traversal for a given Graph.
17. Write a program to represent the given graph using adjacency linked list and implement Breadth-First Search Traversal for a given Graph.

18. Write a program to represent the given graph using adjacency matrix and implement Depth First Search Traversal for a given graph
19. Write a program to represent the given graph using adjacency linked list and implement Depth First Search Traversal for a given graph.
20. Write a program to implement a Hash Table using linear probing as the collision resolution strategy. The table should support the following operations:
- Insert
 - Search
21. Write a program to implement a Hash Table using quadratic probing as the collision resolution strategy. The table should support the following operations:
- Insert
 - Search
22. Implement a Min Heap tree and sort the elements. The following operations should be supported:
- heapify
 - extractMin (Deleting the min element)
 - heapsort
23. Implement the Max Heap tree and sort the elements. The following operations should be supported:
- heapify
 - extractMax (Deleting the max element)
 - heapsort
24. Add 2 non-negative numbers that have been given as a linked list
- Given two non-empty linked lists representing two non-negative integers. The most significant digit comes first and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list. You may assume the two numbers do not contain any leading zero, except the number 0 itself.*
- Example 1:
- operand_1: 7 → 2 → 3 → 3

operand_2: $5 \rightarrow 7 \rightarrow 4$
result: $7 \rightarrow 8 \rightarrow 0 \rightarrow 7$

25. Rotate a Linked List by 'k' places.

Example 1:

input: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$, $k = 1$

output: $5 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$

Example 2:

input: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$, $k = 2$

output: $4 \rightarrow 5 \rightarrow 1 \rightarrow 2 \rightarrow 3$

26. Swap nodes of a linked list in pairs

Example 1:

input: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$

output: $2 \rightarrow 1 \rightarrow 4 \rightarrow 3$

Example 2:

input: $1 \rightarrow 2 \rightarrow 3$

output: $2 \rightarrow 1 \rightarrow 3$

27. Construct a binary tree using recursion and traverse it in all possible traversal.

28. Construct a binary tree and count the number of leaves in the tree

29. Construct a binary tree using recursion and get the height of given node.

30. Construct a Binary search tree and get the minimum and maximum key in the tree.

31. Construct a binary search tree and get the predecessor of the given node.

32. Construct a binary search tree and get the successor of the given node.

33. Construct a binary search tree and get the balance factor of the given node.

34. Construct an AVL tree and execute left rotation on it.

35. Construct an AVL tree and execute right rotation on it.

36.