**1. Write a program in C to create and display Singly Linked List.**

Test Data :

Input the number of nodes : 3

Input data for node 1 : 5

Input data for node 2 : 6

Input data for node 3 : 7

Expected Output :

```
Data entered in the list :
Data = 5
Data = 6
Data = 7
```

**2. Write a program in C to create a singly linked list of n nodes and** display it in reverse order.

Test Data :

Input the number of nodes : 3

Input data for node 1 : 5

Input data for node 2 : 6

Input data for node 3 : 7

Expected Output :

```
Data entered in the list are :
Data = 5
Data = 6
Data = 7
The list in reverse are :
Data = 7
Data = 6
Data = 5
```

**3. Write a program in C to create a singly linked list of n nodes and count the number of nodes.**

Test Data :

Input the number of nodes : 3

Input data for node 1 : 5

Input data for node 2 : 6

Input data for node 3 : 7

Expected Output :

```
Data entered in the list are :
Data = 5
Data = 6
Data = 7

Total number of nodes = 3
```

**4. Write a program in C to insert a new node at the beginning of a Singly Linked List.**

Test Data and Expected Output :

```
Input the number of nodes : 3
Input data for node 1 : 5
Input data for node 2 : 6
Input data for node 3 : 7

Data entered in the list are :
Data = 5
Data = 6
Data = 7

Input data to insert at the beginning of the list : 4


Data after inserted in the list are :
Data = 4
Data = 5
Data = 6
Data = 7
```

**5. Write a program in C to insert a new node at the end of a Singly Linked List.**

Test Data and Expected Output :

```
Input the number of nodes : 3
 Input data for node 1 : 5
 Input data for node 2 : 6
 Input data for node 3 : 7

 Data entered in the list are :
 Data = 5
 Data = 6
 Data = 7

 Input data to insert at the end of the list : 8

 Data, after inserted in the list are :
 Data = 5
 Data = 6
 Data = 7
 Data = 8
```

**6. Write a program in C to insert a new node at the middle of Singly Linked List.**

Test Data and Expected Output :

```
Input the number of nodes (3 or more) : 4
 Input data for node 1 : 1
 Input data for node 2 : 2
 Input data for node 3 : 3
 Input data for node 4 : 4

 Data entered in the list are :
 Data = 1
 Data = 2
 Data = 3
 Data = 4

 Input data to insert in the middle of the list : 5
 Input the position to insert new node : 3

 Insertion completed successfully.

 The new list are :
 Data = 1
 Data = 2
 Data = 5
 Data = 3
 Data = 4
```

**7. Write a program in C to delete first node of Singly Linked List.**

**Test Data :**

Input the number of nodes : 3

Input data for node 1 : 2

Input data for node 2 : 3

Input data for node 3 : 4

Expected Output :

```
 Data entered in the list are :
 Data = 2
 Data = 3
 Data = 4

 Data of node 1 which is being deleted is :  2

 Data, after deletion of first node :
 Data = 3
 Data = 4
```

**8. Write a program in C to delete a node from the middle of Singly Linked List.**

Test Data and Expected Output :

```
Input the number of nodes : 3
 Input data for node 1 : 2
 Input data for node 2 : 5
 Input data for node 3 : 8

 Data entered in the list are :
 Data = 2
 Data = 5
 Data = 8

 Input the position of node to delete : 2

 Deletion completed successfully.

 The new list are  :
 Data = 2
 Data = 8
```

**9. Write a program in Java to delete the last node of Singly Linked List.**

Test Data :

Input the number of nodes : 3

Input data for node 1 : 1

Input data for node 2 : 2

Input data for node 3 : 3

Expected Output :

```
 Data entered in the list are :
 Data = 1
 Data = 2
 Data = 3

 The new list after deletion the last node are  :
 Data = 1
 Data = 2
```

**10. Write a program in C to search an existing element in a singly linked list.**

Test Data and Expected Output :

Input the number of nodes : 3

```
Input data for node 1 : 2
Input data for node 2 : 5
Input data for node 3 : 8

Data entered in the list are :
Data = 2
Data = 5
Data = 8
Input the element to be searched : 5
Element found at node 2
```

**11. Write a program in C to create and display a doubly linked list.**

Test Data :

Input the number of nodes : 3

Input data for node 1 : 2

Input data for node 2 : 5

Input data for node 3 : 8

Expected Output :

```
Data entered on the list are :
node 1 : 2
node 2 : 5
node 3 : 8
```

**12. Write a program in C to create a doubly linked list and display in reverse order.**

Test Data :

Input the number of nodes : 3

Input data for node 1 : 2

Input data for node 2 : 5

Input data for node 3 : 8

Expected Output :

```
Data in reverse order are :
Data in node 1 : 8
Data in node 2 : 5
Data in node 3 : 2
```

**13. Write a program in C to insert a new node at the beginning in a doubly linked list.**

Test Data and Expected Output :

```
Input the number of nodes : 3
Input data for node 1 : 2
Input data for node 2 : 5
Input data for node 3 : 8

Data entered in the list are :
node 1 : 2
node 2 : 5
node 3 : 8
Input data for the first node : 1

After insertion the new list are :
node 1 : 1
node 2 : 2
node 3 : 5
node 4 : 8
```

**14. Write a program in C to insert a new node at the end of a doubly linked list.**

Test Data and Expected Output :

```
Input the number of nodes : 3
 Input data for node 1 : 2
 Input data for node 2 : 5
 Input data for node 3 : 8

 Data entered in the list are :
 node 1 : 2
 node 2 : 5
 node 3 : 8
 Input data for the last node : 9

 After insertion the new list are :
 node 1 : 2
 node 2 : 5
 node 3 : 8
 node 4 : 9
```

**15. Write a program in C to delete a node from the last of a doubly linked list.**

Test Data and Expected Output :

```
Input the number of nodes (3 or more ): 3
 Input data for node 1 : 1
 Input data for node 2 : 2
 Input data for node 3 : 3

 Data entered in the list are :
 node 1 : 1
 node 2 : 2
 node 3 : 3

 After deletion the new list are :
 node 1 : 1
 node 2 : 2
```

**16. Write a program in C to create and display a circular linked list.**

Test Data :

```
Input the number of nodes : 3
Input data for node 1 : 2
Input data for node 2 : 5
```

Input data for node 3 : 8

Expected Output :

Data entered in the list are :

Data 1 = 2

Data 2 = 5

Data 3 = 8

**17. Write a C programming to sort a given linked list.**

Test Data and Expected Output : 5

15

33

49

6

65

Input number of elements in the linked list? Input the elements in the linked list:

Sorted order is:

```
6        15       33       49       65
```

---

**STACK Related Program**

================================================================

**Q18. Write a C program to implement a stack using a singly linked list.**

Expected Output:

```
 Push data 1

 Push data 2

 Push data 3

 Push data 4



 Pop data: 4

 Pop data: 3

 Pop data: 2

 Pop data: 1

 Check a stack is empty or not?

 Stack is empty!
```

Q19. **Write a C program to check a stack is full or not using an array with push and pop operations. >**

Expected Output:

```
Stack size: 3
Original Stack: 1 2 3
Push another value and check if the stack is full or not!
Stack is full!

Stack size: 3
Original Stack: 10 20
Check the said stack is full or not!
Stack is not full!
```

Q20. **Write a C program that checks whether a string of parentheses is balanced or not using stack**.

Expected Output:

```
Input an expression in parentheses: {[])
The expression is not balanced.
 ----------------------------------------
 Input an expression in parentheses: ((()))
The expression is balanced.
 ----------------------------------------
 Input an expression in parentheses: ())
The expression is not balanced.
 ----------------------------------------
 Input an text of parentheses: ([]){}[[(){}]{}]
The expression is balanced.
 ----------------------------------------
 Input an expression in parentheses: [(]))
The expression is not balanced.
```

**Q21. Write a C program to find the next greater element for each element in an array using a stack. Return -1 if there is no next-larger element.**

**Expected Output:**

```
Elements in the array are: 1 2 3 4 5 6

The next larger elements are:
1 --> 2
2 --> 3
3 --> 4
4 --> 5
5 --> 6
6 --> -1

Elements in the array are: 6 5 4 3 2 1 0
The next larger elements are:
0 --> -1
1 --> -1
2 --> -1
3 --> -1
4 --> -1
5 --> -1
6 --> -1

Elements in the array are: 3 7 5 9 3 2 4 1 4
The next larger elements are:
3 --> 7
5 --> 9
7 --> 9
2 --> 4
3 --> 4
1 --> 4
4 --> -1
9 --> -1
```

**Q22. Write a C program to implement two stacks using a single array.**
**Expected Output:**

```
3 popped from stack 1
2 popped from stack 1
1 popped from stack 1

30 popped from stack 2
20 popped from stack 2
10 popped from stack 2
```

**Q23. Write a C program that reverses a stack using only stack operations push and pop.**
Expected Output:

```
Original Stack: 10 20 30 40 50
Reversed Stack: 50 40 30 20 10
```

**Q24. Write a C program to find the minimum element in a stack.**

Expected Output:

```
Current stack elements:

9 2 4 2 4
Minimum element: 2

After removing two elements:
Current stack elements:
9 2 4
Minimum element: 2

After adding one element:
Current stack elements:
9 2 4 1
Minimum element: 1
```

**Q.25 Write a C program to find the maximum element in a stack. >**
Expected Output:

```
Current stack elements:
5 2 1 6 8
Maximum element: 8

After removing two elements:
Current stack elements:
5 2 1
Maximum element: 5

After adding one element:
Current stack elements:
5 2 1 10
Maximum element: 10
```

**Q26. Write a C program to implement a stack that supports push, pop, get middle, and delete middle elements. >**
Expected Output:

```
Stack elements: 88 15 26 32 23
Middle element: 26

Delete the middle element of the said stack:
Stack elements: 88 15 32 23
Middle element: 15

Delete the middle element of the said stack:
Stack elements: 88 32 23
Middle element: 32
```

**Q27. Write a C program to implement a stack and accept some numeric values. Remove the number whose value is the minimum on the stack. >**

Expected Output:

```
Elements of the stack:
Stack: 7 4 5 2 3 1
Minimum value of the said stack: 1
Elements of the stack after removing the said minimum value:
Stack: 7 4 5 2 3
Minimum value of the said stack: 2
Elements of the stack after removing the said minimum value:
Stack: 7 4 5 3
Minimum value of the said stack: 3
Elements of the stack after removing the said minimum value:
Stack: 7 4 5
```

**Q28. Write a C program to implement a stack and accept some numeric values. Find the top and kth element of the stack. >**

Expected Output:

```
Elements of the stack:
1 2 3 4 5 6
Top element: 6
3rd element from top: 4
Remove the topmost element from the stack:
Elements of the stack:
1 2 3 4 5
Top element: 5
4th element from top: 2
```

Q29. Write a C program to convert a decimal number to its binary equivalent using stack. >
Expected Output:

```
Input a decimal number: 10
The binary equivalent is: 1010
Input a decimal number: 109
Binary equivalent of the said number is: 1101101

Input a decimal number: 2015
Binary equivalent of the said number is: 11111011111
```

==========================================================

## QUEUE REALTED PROGRAM

**Q30. Write a C program to implement a queue using an array. Programs should contain functions for inserting elements into the queue, displaying queue elements, and checking whether the queue is empty or not.**

Expected Output:

```
Initialize a queue!
Check the queue is empty or not? Yes

Insert some elements into the queue:
Queue elements are: 1 2 3

Insert another element into the queue:
Queue elements are: 1 2 3 4

Check the queue is empty or not? No
```

**Q31. Write a C program to implement a queue using an array. Create a function that removes an element from the queue.**

Expected Output:

```
Initialize a queue!

Insert some elements into the queue:

Queue elements are: 1 2 3



Delete an element from the said queue:

Queue elements are: 2 3



Insert another element into the queue:
```

```
Queue elements are: 2 3 4
```

**Q32.Write a C program to implement a queue using a linked list. Programs should contain functions for inserting elements into the queue, displaying queue elements, and checking whether the queue is empty or not. >**

Expected Output:

```
Initialize a queue!
Check the queue is empty or not? Yes

Insert some elements into the queue:
1 2 3

Insert another element into the queue:
1 2 3 4

Check the queue is empty or not? No
```

**Q33. Write a C program to count the number of elements in a queue.**

Expected Output:

```
Initialize a queue!
```

```
Check the queue is empty or not? Yes
Number of elements in queue: 0

Insert some elements into the queue:
Queue elements are: 1 2 3
Number of elements in queue: 3

Delete two elements from the said queue:
Queue elements are: 3
Number of elements in queue: 1

Insert another element into the queue:
Queue elements are: 3 4
Number of elements in the queue: 2
```

**Q34. Write a C program to reverse the elements of a queue. >**
Expected Output:

```
Queue elements are:
1 2 3 4 5
Reverse Queue, elements are:
5 4 3 2 1
Add two elements to the said queue:
Queue elements are:
5 4 3 2 1 100 200
Reverse Queue, elements are:
200 100 1 2 3 4 5
```

**Q35. Write a C program to calculate the sum of the elements in a queue. >**
Expected Output:

```
Queue elements are: 1 2 3 4 5
Sum of the elements in the queue is: 15

Remove 2 elements from the said queue:
Queue elements are: 3 4 5
Sum of the elements in the queue is: 12

Insert 3 more elements:
Queue elements are: 3 4 5 300 400 500
Sum of the elements in the queue is: 1212
```

**Q36. Write a C program to compute the average value of the elements in a queue. >**
Expected Output:

```
Queue elements are: 1 2 3 4 5
Average of the elements in the queue is: 3.000000

Remove 2 elements from the said queue:
Queue elements are: 3 4 5
Average of the elements in the queue is: 4.000000

Insert 3 more elements:
Queue elements are: 3 4 5 300 427 519
Average of the elements in the queue is: 209.666672
```

**Q37. Write a C program to find the maximum element in a queue. >**
Expected Output:

```
Queue elements are: 1 2 3 4 5
Maximum value in the queue is: 5

Remove 2 elements from the said queue:
Queue elements are: 3 4 5
Maximum value in the queue is: 5

Insert 3 more elements:
Queue elements are: 3 4 5 600 427 519
Maximum value in the queue is: 600
```

**38. Write a C program to find the minimum element in a queue. >**
Expected Output:

```
Queue elements are: 1 2 3 4 5
Minimum value in the queue is: 1

Remove 2 elements from the said queue:
Queue elements are: 3 4 5
Minimum value in the queue is: 3

Insert 3 more elements:
Queue elements are: 3 4 5 600 -427 519
Minimum  value in the queue is: -427
```

Q39. **Write a C program to delete the nth element of a queue. > Firstly, this code checks if the queue is empty or if the position to delete is invalid. If the position is valid, it deletes the first n-1**

**elements, then deletes the nth element by calling the dequeue() function.**

Expected Output:

```
Insert some elements into the queue:
Queue elements are: 1 2 3 4 5

Delete the 7th element of the said queue:
Error: Invalid position

Delete the 3rd element of the said queue:
Queue elements are: 4 5
```

**Q40. Write a C program to sort the elements of a queue in ascending order. >**

**Expected Output:**

```
Input some elements into the queue:
Elements of the queue:
4 2 7 5 1

Sort the said queue:
Elements of the sorted queue in ascending order:
1 2 4 5 7

Input two more elements into the queue:
Elements of the queue:
1 2 4 5 7 -1 3

Sort the said queue:
Elements of the sorted queue in ascending order:
-1 1 2 3 4 5 7
```

**Q41. Write a C program to find the median of the elements in a queue. >**
**From Wikipedia,**
**In statistics and probability theory, the median is the value separating the higher half from the lower half of a data sample, a population, or a probability distribution.**
**The median of a finite list of numbers is the "middle" number, when those numbers are listed in order from smallest to greatest.**
**If the data set has an odd number of observations, the middle one is selected. For example, the following list of seven numbers,**
**1, 3, 3, 6, 7, 8, 9 has the median of 6, which is the fourth value.**
**If the data set has an even number of observations, there is no distinct middle value and the median is usually defined to be the arithmetic mean of the two middle values. For example, this data set of 8 numbers1, 2, 3, 4, 5, 6, 8, 9 has a median value of 4.5, that is (4+5)/2 . (In more technical terms, this interprets the median as the fully trimmed mid-range).**
Expected Output:

```
Input some elements into a queue:

Queue elements are:

1 2 3 4 5

The median of the elements in the queue is 3.000000

Input one more element:

Queue elements are:

1 2 3 4 5 6

The median of the elements in the queue is 3.500000
```

**Q42. Write a C program to find the median of the elements in a queue. >**

Expected Output:

```
Input some elements into a queue:
Queue elements are:
1 2 3 4 5
The median of the elements in the queue is 3.000000

Input one more element:
Queue elements are:
1 2 3 4 5 6
The median of the elements in the queue is 3.500000
```

Tree Related Program

Q43. Write a Program to Calculate Size of a Tree | Recursion

Q44. Find the Node with Minimum Value in a Binary Search Tree

Q45. Program to Determine if Given Two Trees are Identical or Not

Q46. Find the Maximum Depth or Height of Given Binary Tree

Q47. Print Nodes at K Distance from Root

Q48. Program to Count Leaf Nodes in a Binary Tree

Q49. Lowest Common Ancestor in a Binary Search Tree.

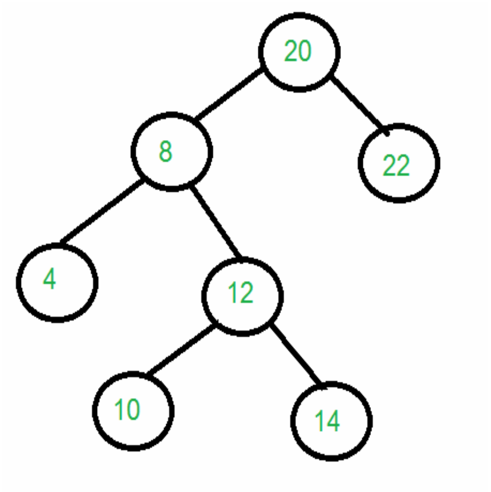
Q50. A Program to Check if a Binary Tree is BST Or Not

Q51. Find K-th Smallest Element in BST (Order Statistics in BST)

Q52. Inorder Tree Traversal without Recursion and Without Stack!

Q53. Print All Nodes at Distance K from a Given Node

Given a binary tree, a target node in the binary tree, and an integer value k, print all the nodes that are at distance k from the given target node. No parent pointers are available.



Consider the tree shown in diagram

Input: target = pointer to node with data 8.

root = pointer to node with data 20.

k = 2.

Output : 10 14 22

If target is 14 and k is 3, then output

should be “4 20”