**Table of Contents**

[**Program 1 Array operations 2**](#_heading=h.uaxy48igdvua)

[**Program 2 Employee Structure 8**](#_heading=h.ixii5idml88t)

[**Program 3 Stack Implementation 11**](#_heading=h.qz5o20lsv1q9)

[**Program 4 Infix to Postfix Conversion 14**](#_heading=h.dmui45ir24vx)

[**Program 5 Evaluation of Postfix Expression 17**](#_heading=h.h0ypfq6m5rau)

[**Program 6 Recursion applications 21**](#_heading=h.h6a19oglnizb)

[6a. Binary Search 21](#_heading=h.cbagezzglu8n)

[6b. Tower of Hanoi problem 23](#_heading=h.9qeu9ox8qk0w)

[**Program 7 Call holding in call center 24**](#_heading=h.4vuedvylzncn)

[**Program 8 Circular Queue using Structure 29**](#_heading=h.kwp6cow9mkai)

[**Program 9 Linked list for sorted names 34**](#_heading=h.ha3wgexoeaku)

[**Program 10 Stack using Linked List 38**](#_heading=h.6or39mnw57kk)

[**Program 11 Doubly Linked List implementation 43**](#_heading=h.hwxlt1iajky6)

[**Program 12 Binary Search Tree implementation 48**](#_heading=h.iw9wc6p4oom5)

# Program 1 Array operations

Design, Develop and Implement a menu driven Program in C for the

following array operations.

a) Creating an array of N Integer Elements

b) Display of array Elements with Suitable Headings

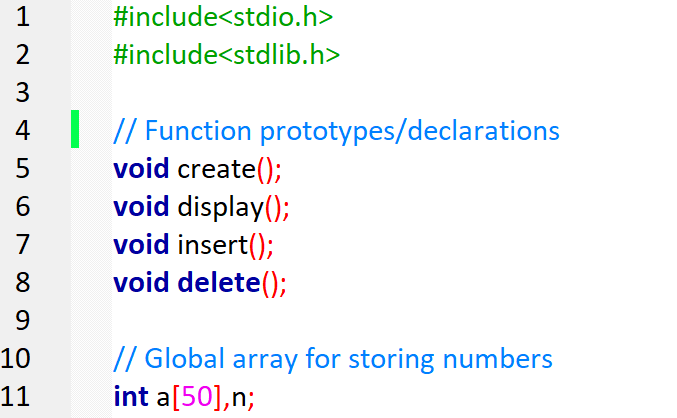
c) Inserting an Element (ELEM) at a given valid Position (POS)

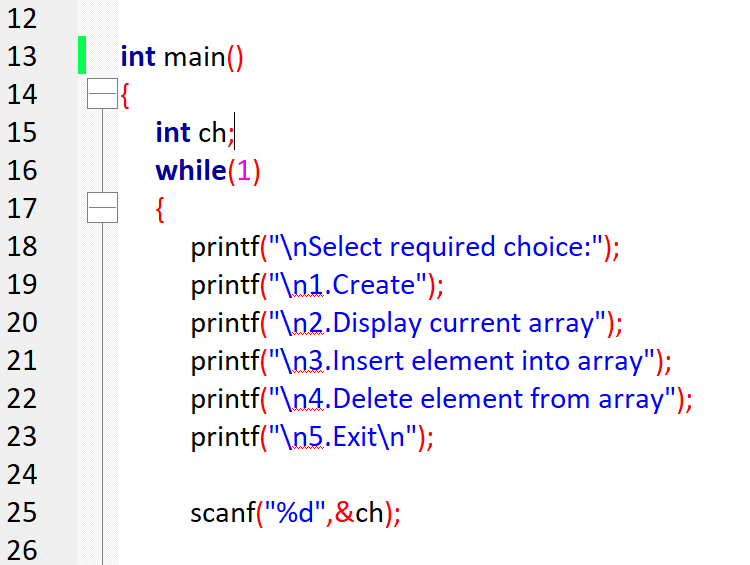
d) Deleting an Element at a given valid Position(POS)

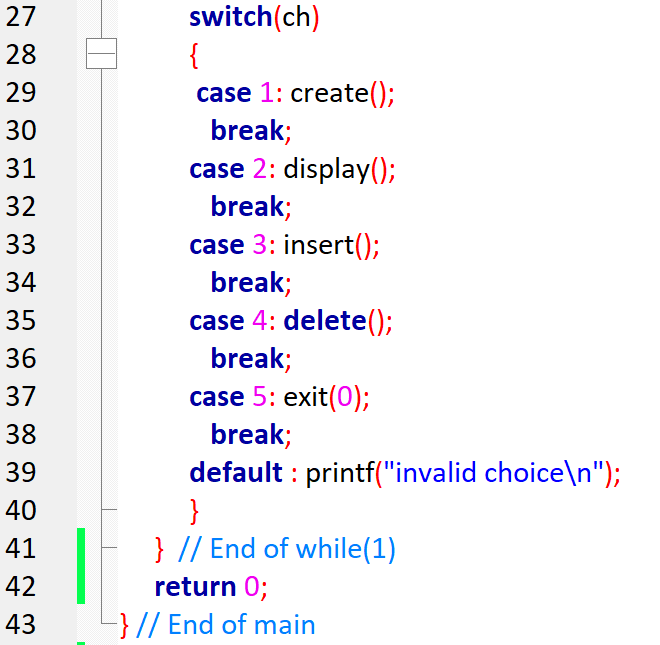
e) Exit.

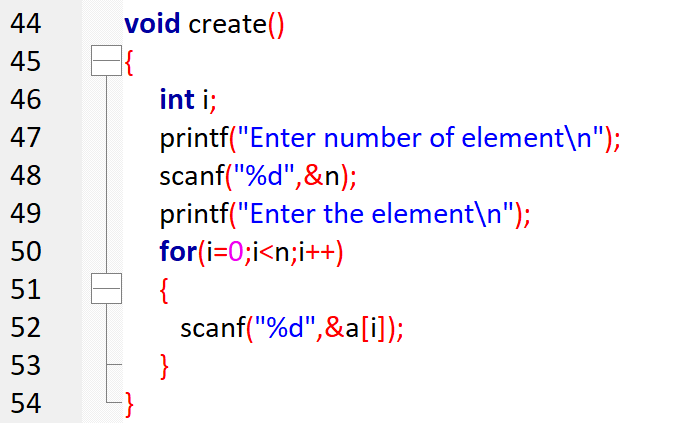
Support the program with functions for each of the above operations.

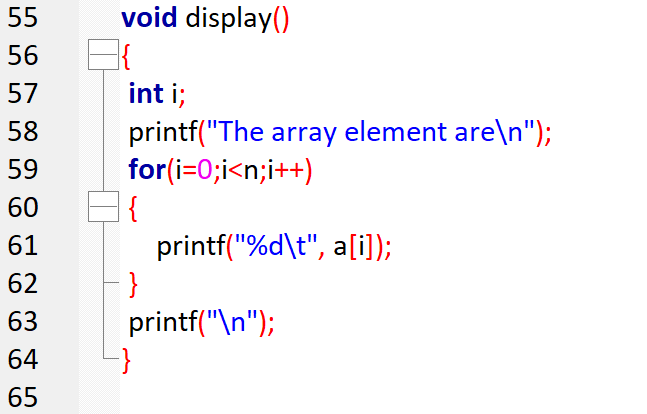
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**

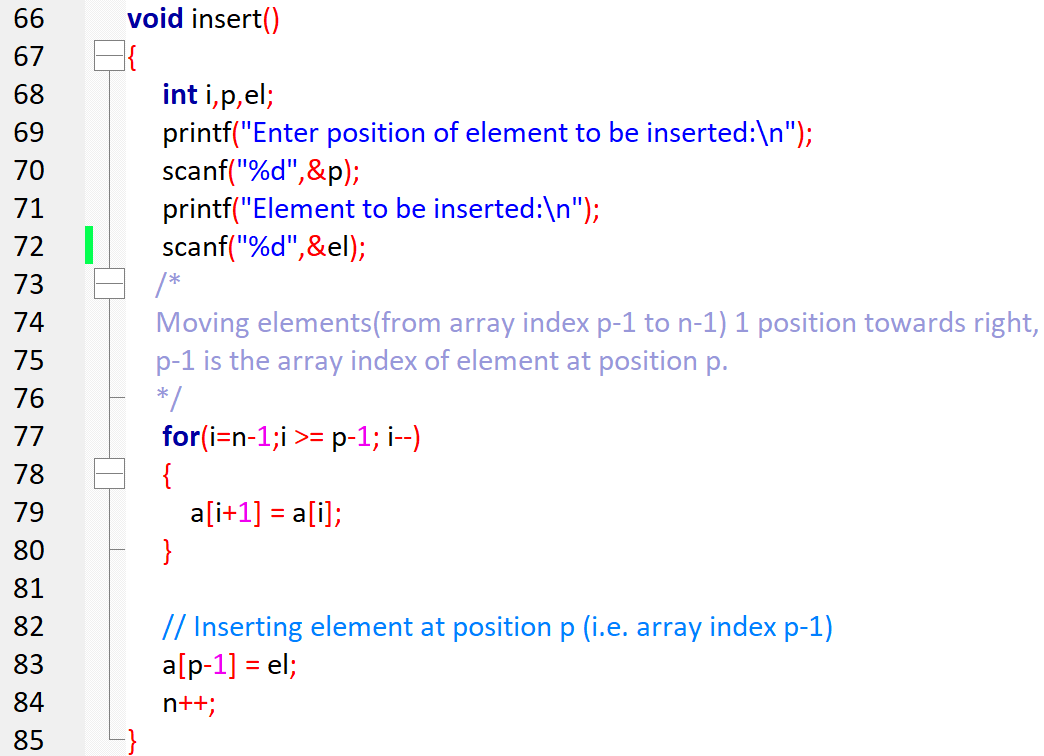


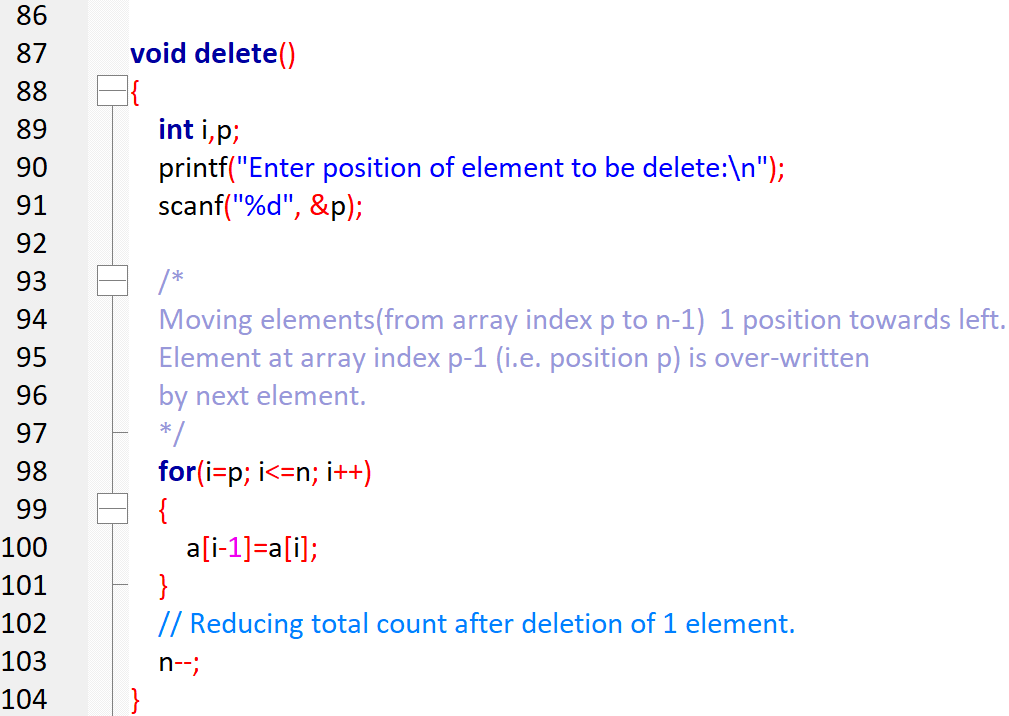












**OUTPUT:**

Select required choice:

1.Create

2.Display current array

3.Insert element into array

4.Delete element from array

5.Exit

1

Enter number of element

4

Enter the element

10

20

30

40

Select required choice

1.Create

2.Display current array

3.insert element into array

4.Delete element from array

5.Exit

2

the array element are

10 20 30 40

select required choice

1.Create

2.Display current array

3.insert element into array

4.Delete element from array

5.Exit

3

Enter position of element to be inserted

2

element to be inserted

50

select required choice

1.Create

2.Display current array

3.insert element into array

4.Delete element from array

5.Exit

2

the array element are

10 50 20 30 40

select required choice

1.Create

2.Display current array

3.insert element into array

4.Delete element from array

5.Exit

4

Enter position of element to be delete

5

select required choice

1.Create

2.Display current array

3.insert element into array

4.Delete element from array

5.Exit

2

the array element are

10 50 20 30

select required choice

1.Create

2.Display current array

3.insert element into array

4.Delete element from array

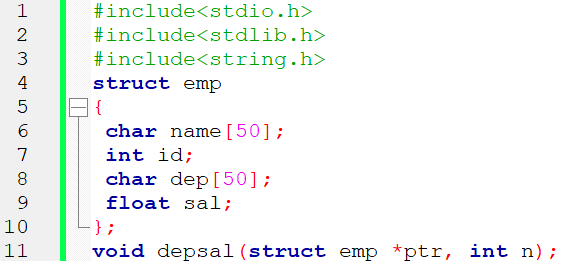
5.Exit

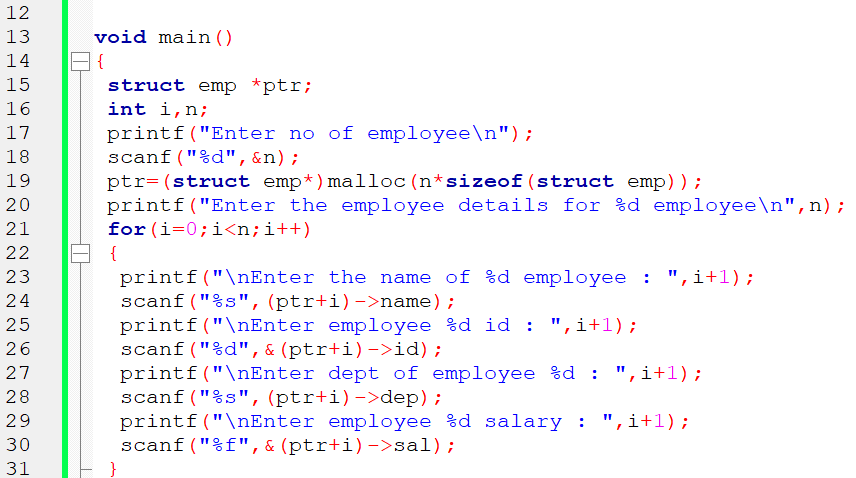
# 

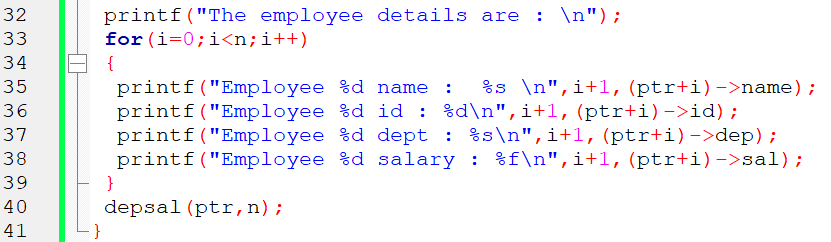
# Program 2 Employee Structure

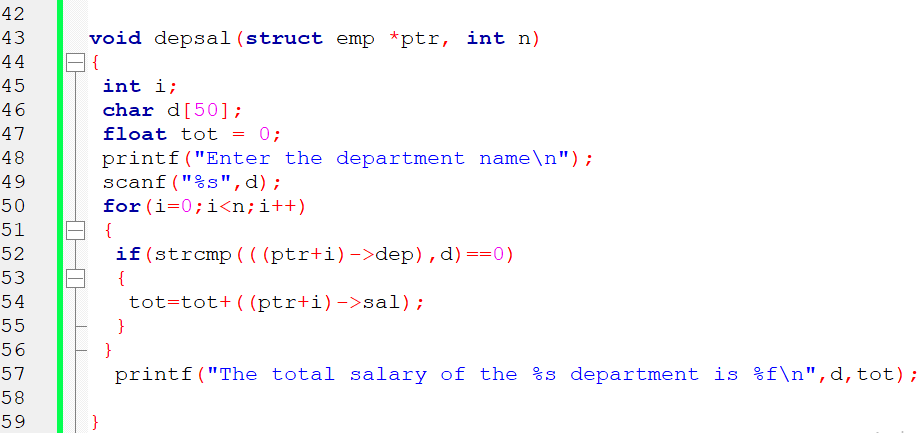
Define an EMPLOYEE structure with members Emp\_name, Emp-id, Dept-name and Salary. Read and display data of N employees. Employees may belong to different departments. Write a function to find total salary of employees of a specified department. Use the concept of pointer to structure and allocate the memory dynamically to EMPLOYEE instances.

**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**









**OUTPUT**

Enter no of employee

3

Enter the employee details for 3 employee

Enter the name of 1 employee : tabraiz

Enter employee 1 id : 10

Enter dept of employee 1 : cse

Enter employee 1 salary : 70000

Enter the name of 2 employee : kshitij

Enter employee 2 id : 20

Enter dept of employee 2 : cse

Enter employee 2 salary : 80000

Enter the name of 3 employee : jeet

Enter employee 3 id : 30

Enter dept of employee 3 : ise

Enter employee 3 salary : 60000

The employee details are :

Employee 1 name : tabraiz

Employee 1 id : 10

Employee 1 dept : cse

Employee 1 salary : 70000.000000

Employee 2 name : kshitij

Employee 2 id : 20

Employee 2 dept : cse

Employee 2 salary : 80000.000000

Employee 3 name : jeet

Employee 3 id : 30

Employee 3 dept : ise

Employee 3 salary : 60000.000000

Enter the department name

cse

the total salary of the cse department is 150000.000000

# Program 3 Stack Implementation

STACK of Integers (Array Implementation of Stack with maximum size MAX)

a) Push an Element on to Stack

b) Pop an Element from Stack

c) Check if stack contents form a palindrome

d) Demonstrate Overflow and Underflow situations on Stack

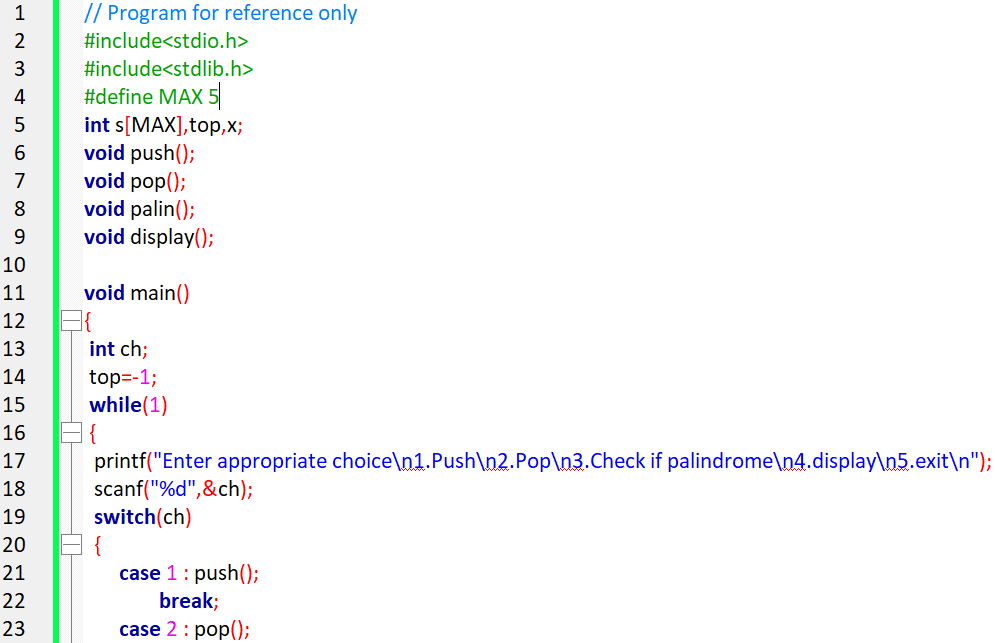
e) Display the status of Stack

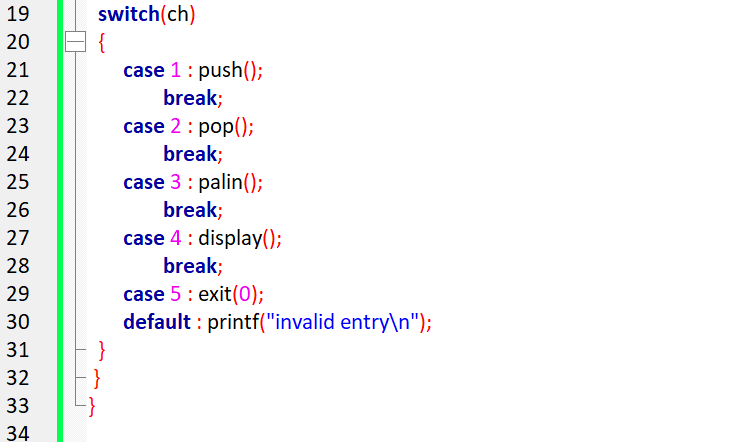
f) Exit

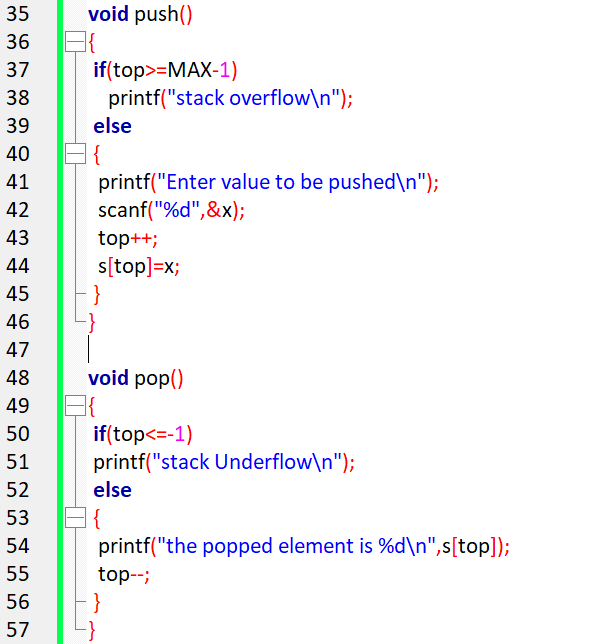
Support the program with appropriate functions for each of the above

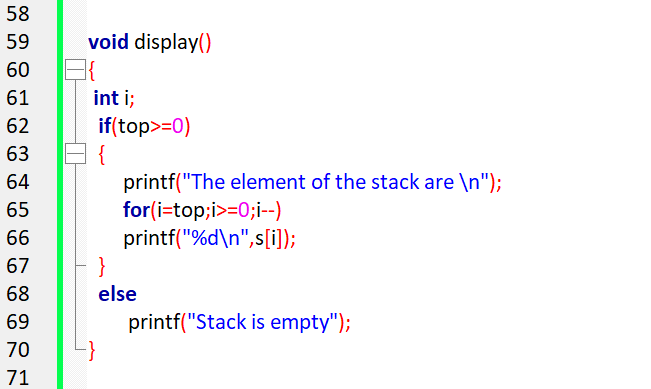
operations

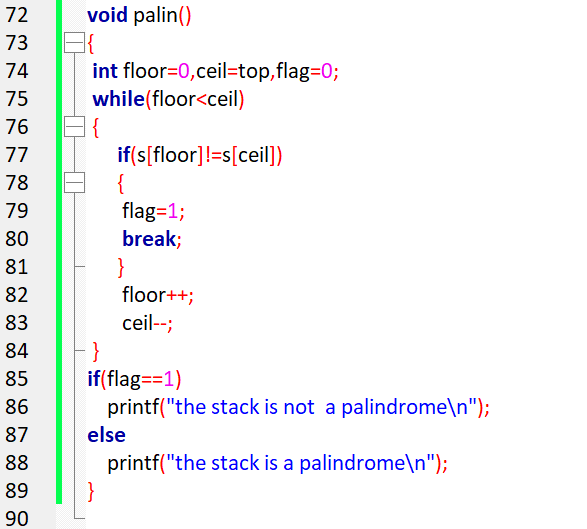
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**









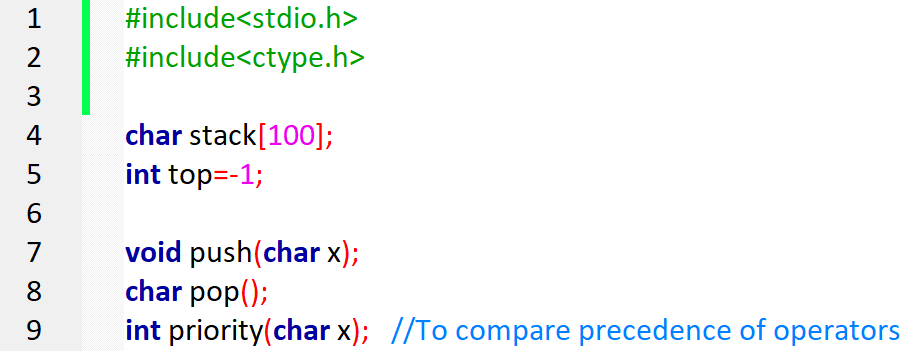


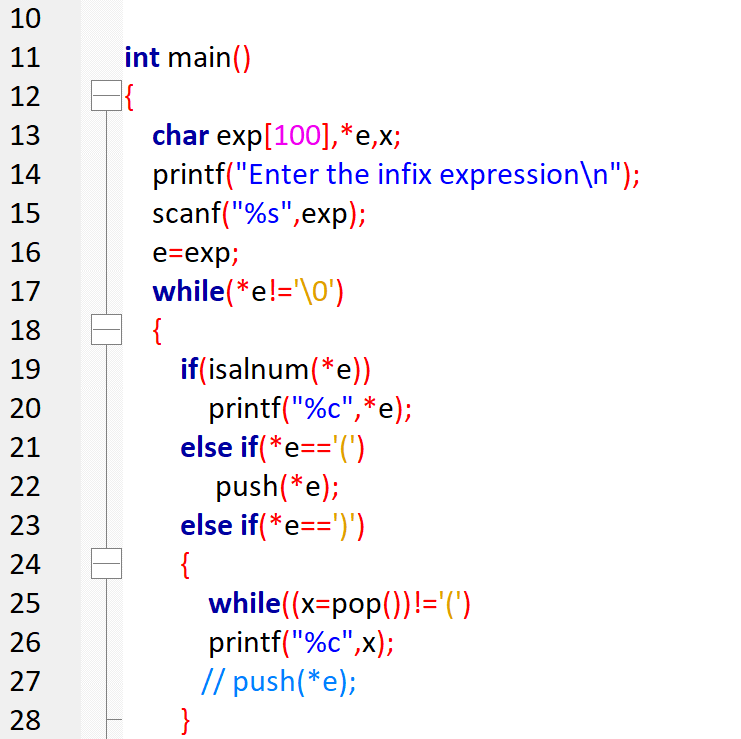
# 

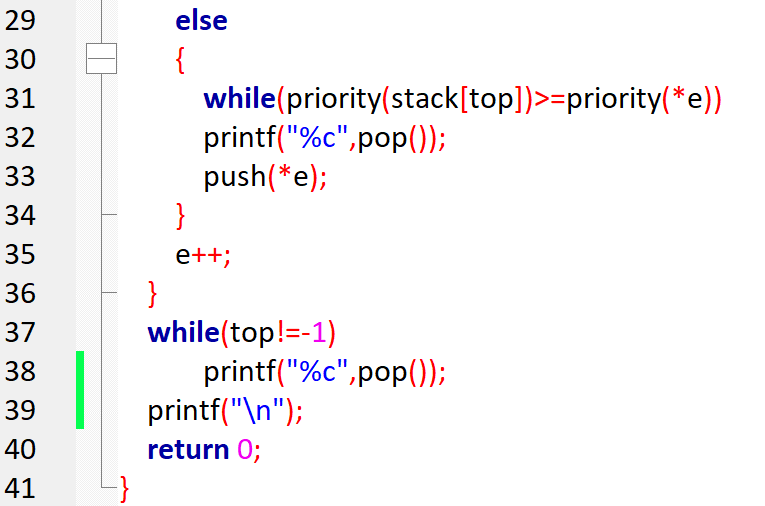
# Program 4 Infix to Postfix Conversion

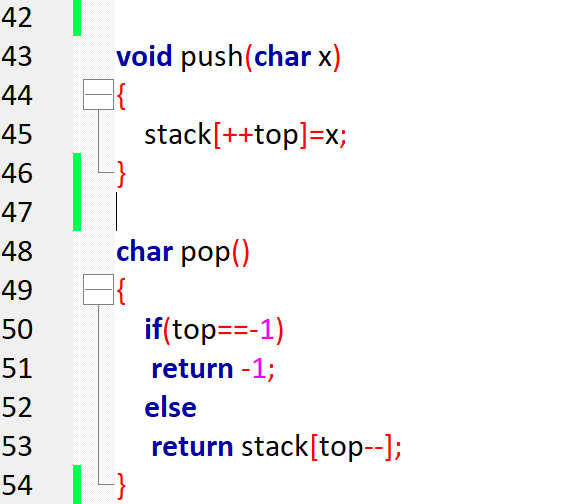
Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and binary operators + - \* /. Apply the concept of stack data structure to solve this problem.

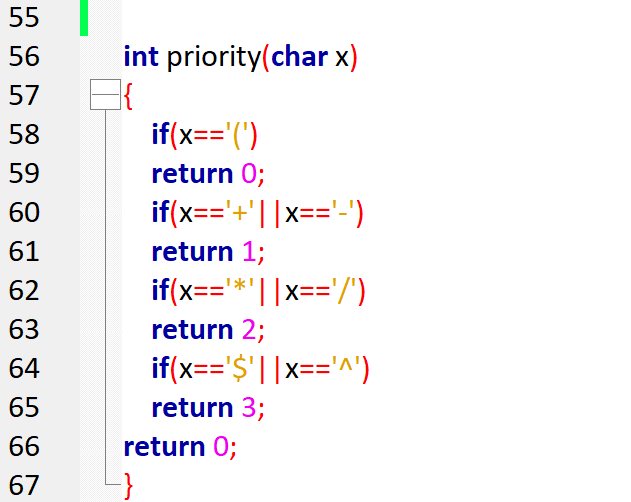
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**











**OUTPUT**

Enter the infix expression

A+B/C\*(D-E)

ABC/DE-\*+

Enter the infix expression

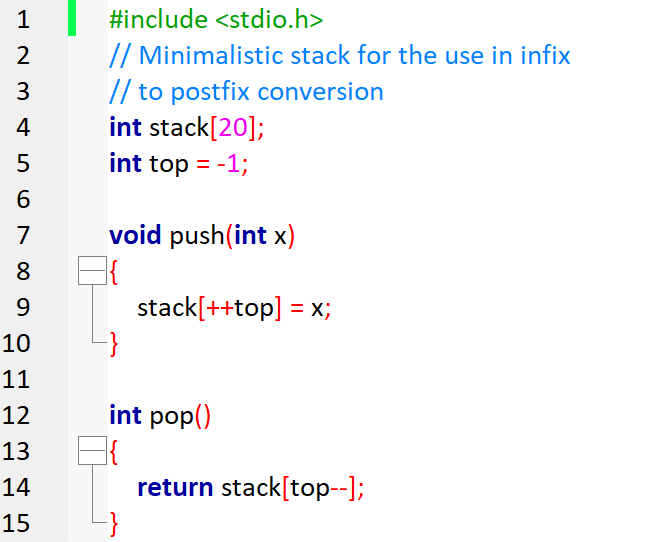
A+B\*C

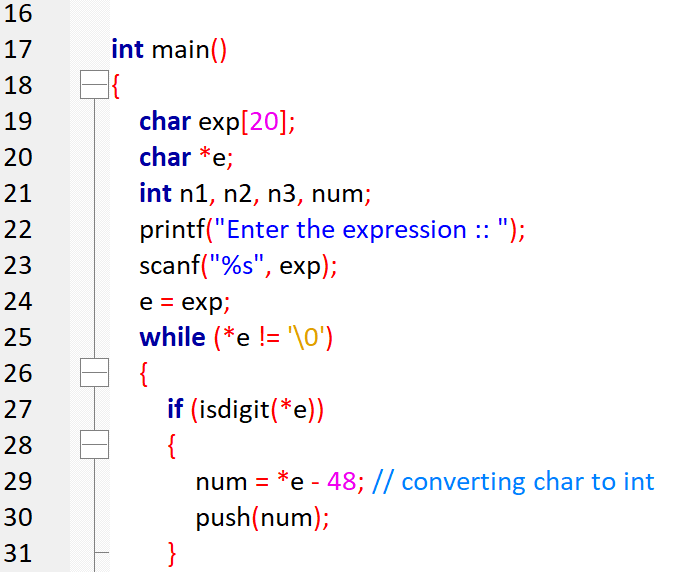
ABC\*+

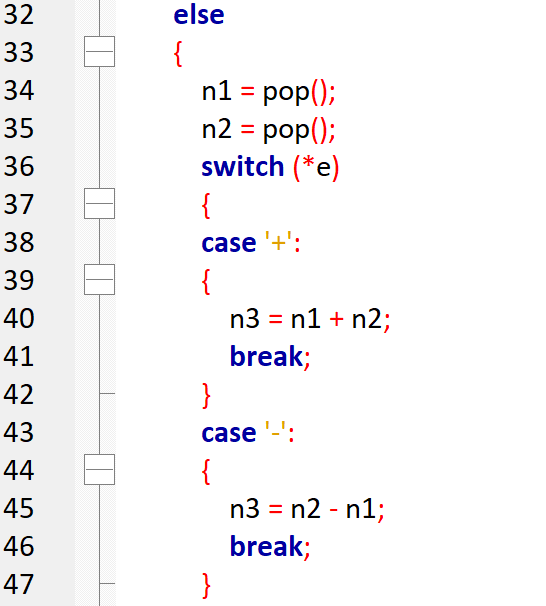
# Program 5 Evaluation of Postfix Expression

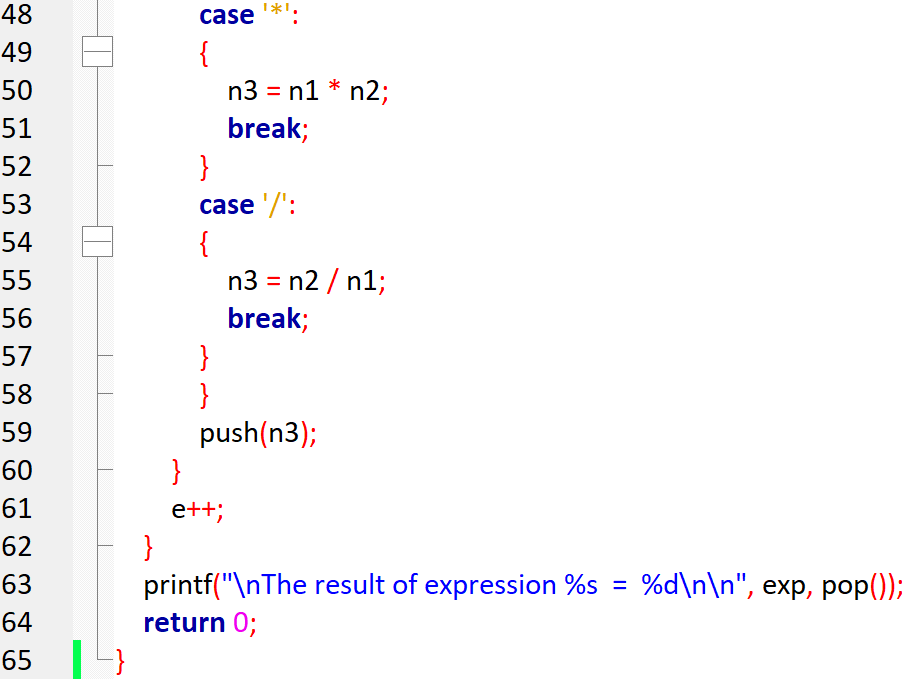
Write a C program to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary operators. The operators are + - \* and /.

**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**









**OUTPUT**

Enter the postfix expression

53+62/\*35\*+

Thes result of 53+62/\*35\*+ = 39

# Program 6 Recursion applications

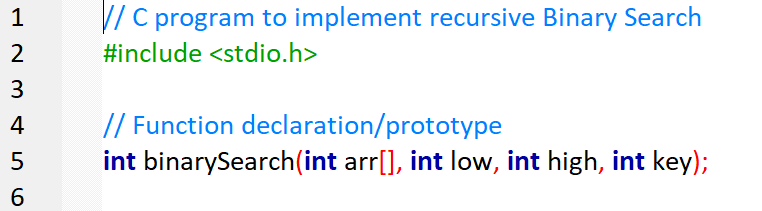
Write recursive functions for the following and demonstrate their use.

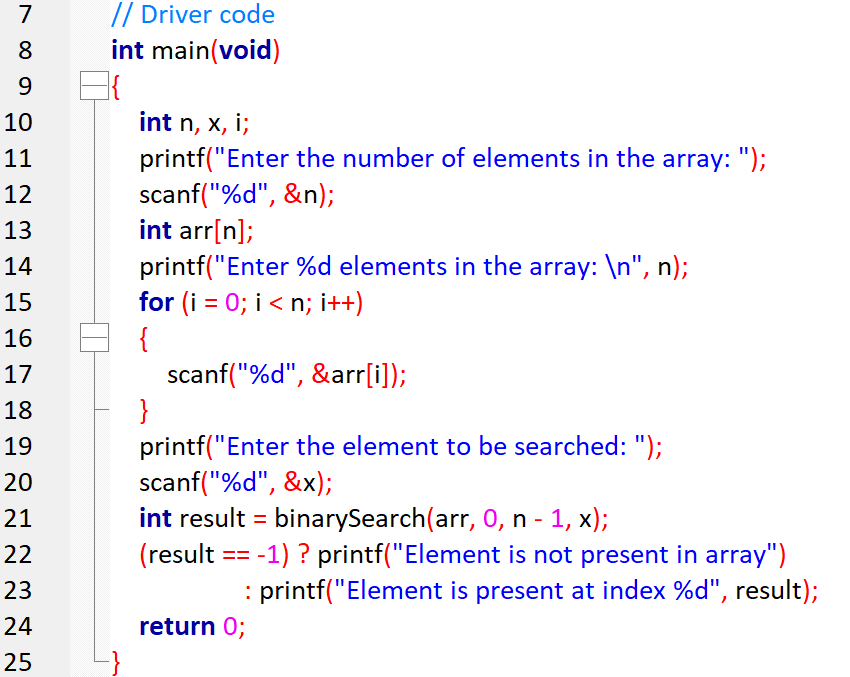
a) Binary Search

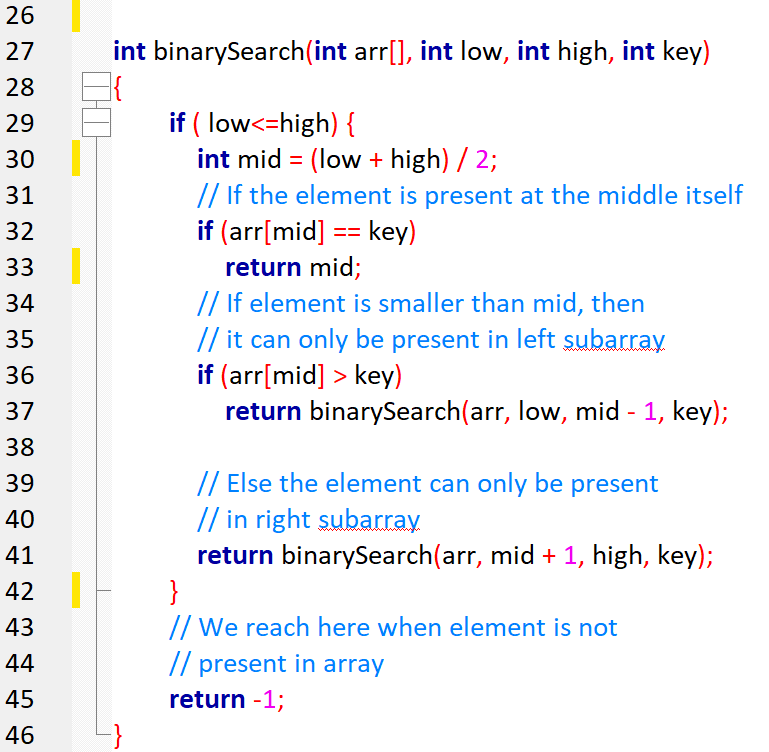
b) Tower of Hanoi problem

**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**

## 6a. Binary Search







**OUTPUT:**

Enter the number of elements in the array: 5

Enter 5 elements in the array:

1 4 7 9 12

Enter the element to be searched: 9

Element is present at index 3

Enter the number of elements in the array: 5

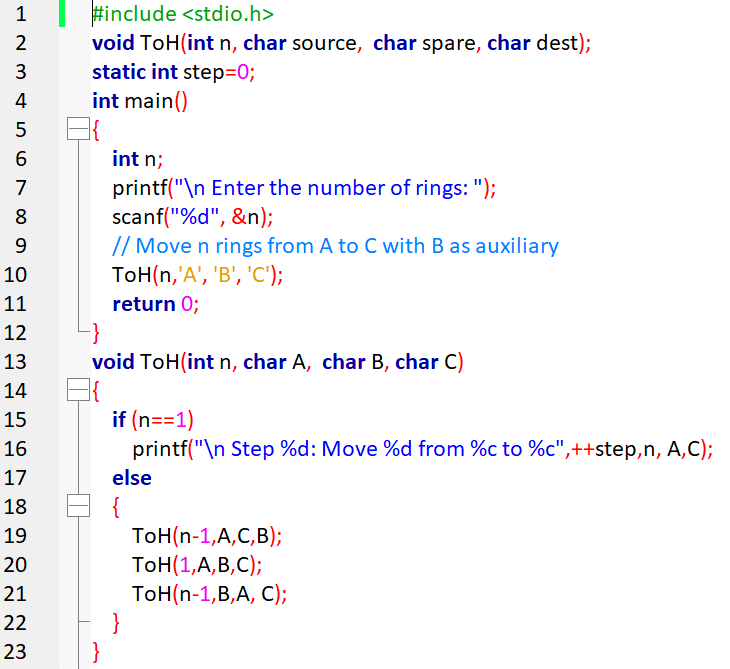
Enter 5 elements in the array:

1 4 7 9 12

Enter the element to be searched: 15

Element is not present in array

## 6b. Tower of Hanoi problem

****

**OUTPUT**

Enter the number of rings: 4

Step 1: Move 1 from A to B

Step 2: Move 1 from A to C

Step 3: Move 1 from B to C

Step 4: Move 1 from A to B

Step 5: Move 1 from C to A

Step 6: Move 1 from C to B

Step 7: Move 1 from A to B

Step 8: Move 1 from A to C

Step 9: Move 1 from B to C

Step 10: Move 1 from B to A

Step 11: Move 1 from C to A

Step 12: Move 1 from B to C

Step 13: Move 1 from A to B

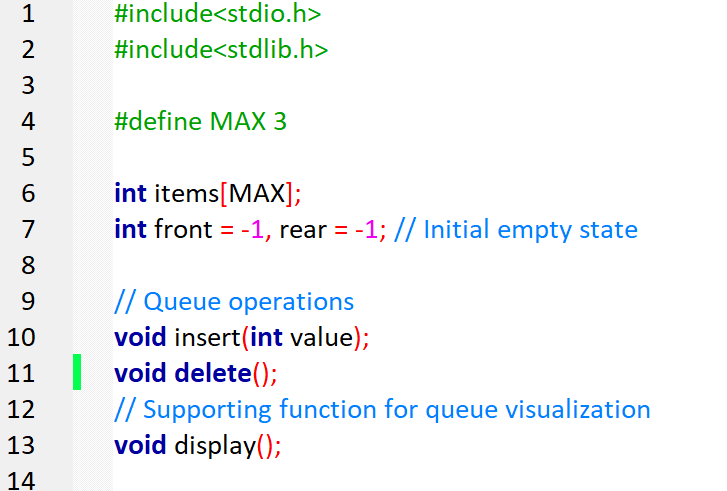
Step 14: Move 1 from A to C

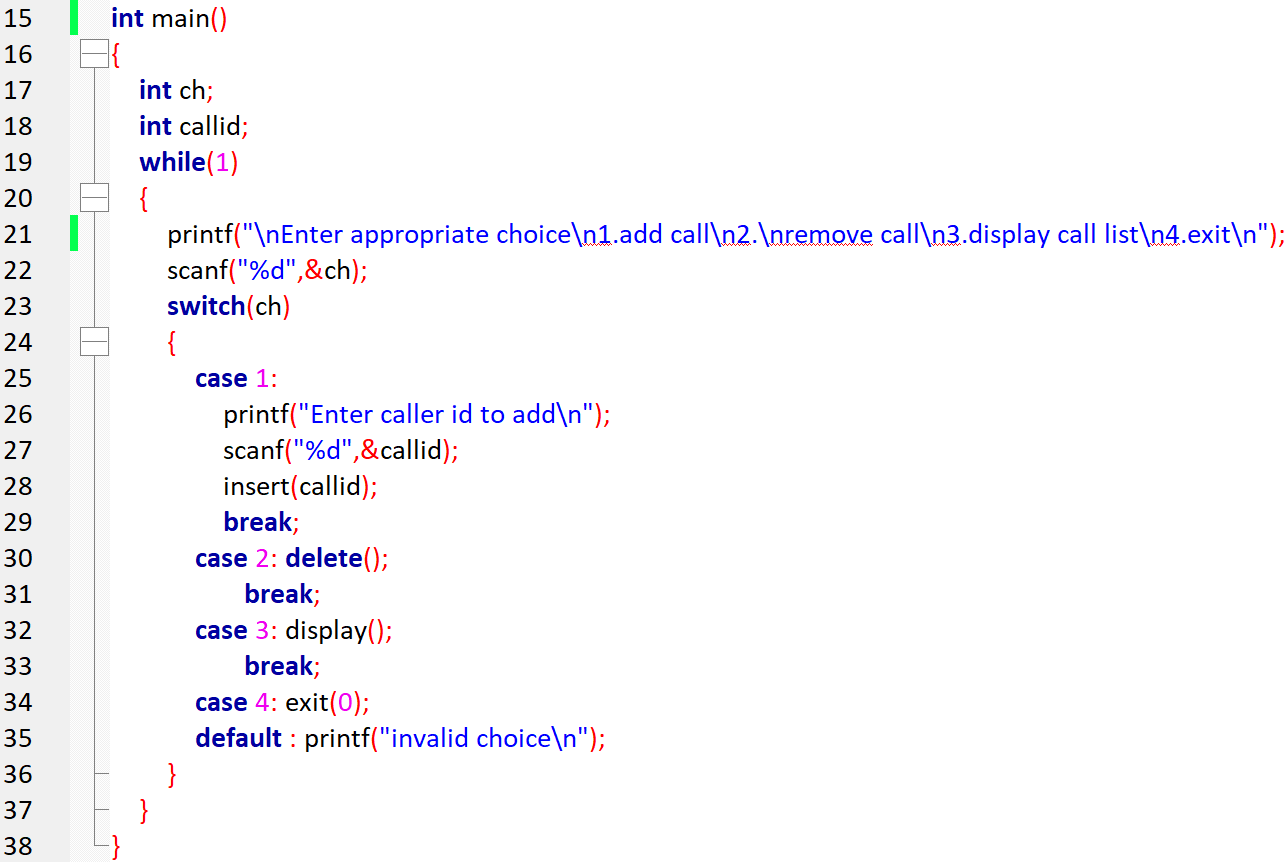
Step 15: Move 1 from B to C

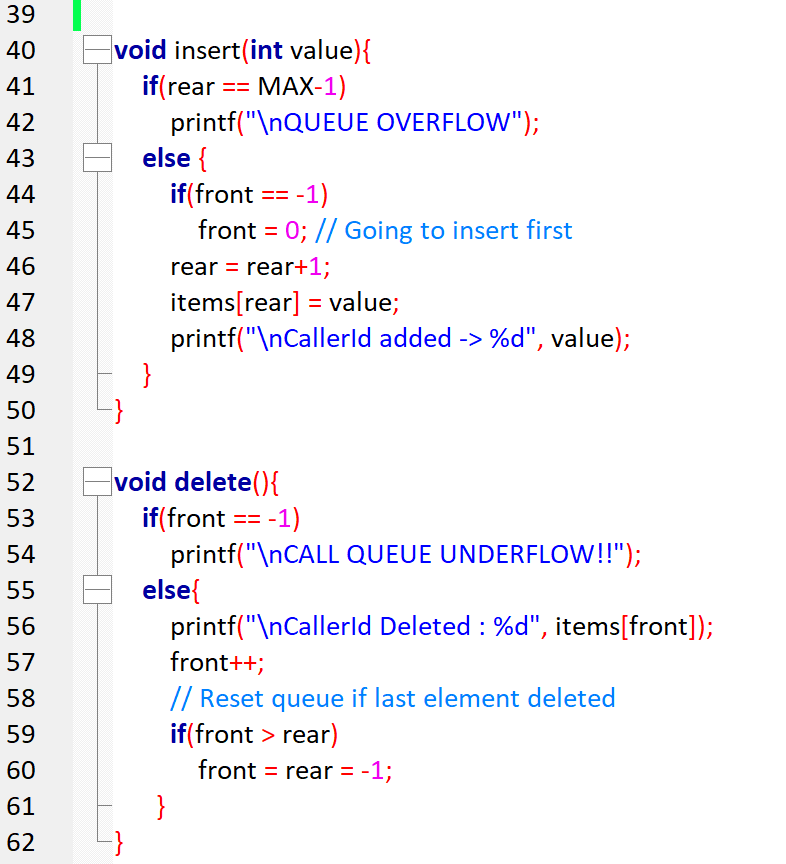
# Program 7 Call holding in call center

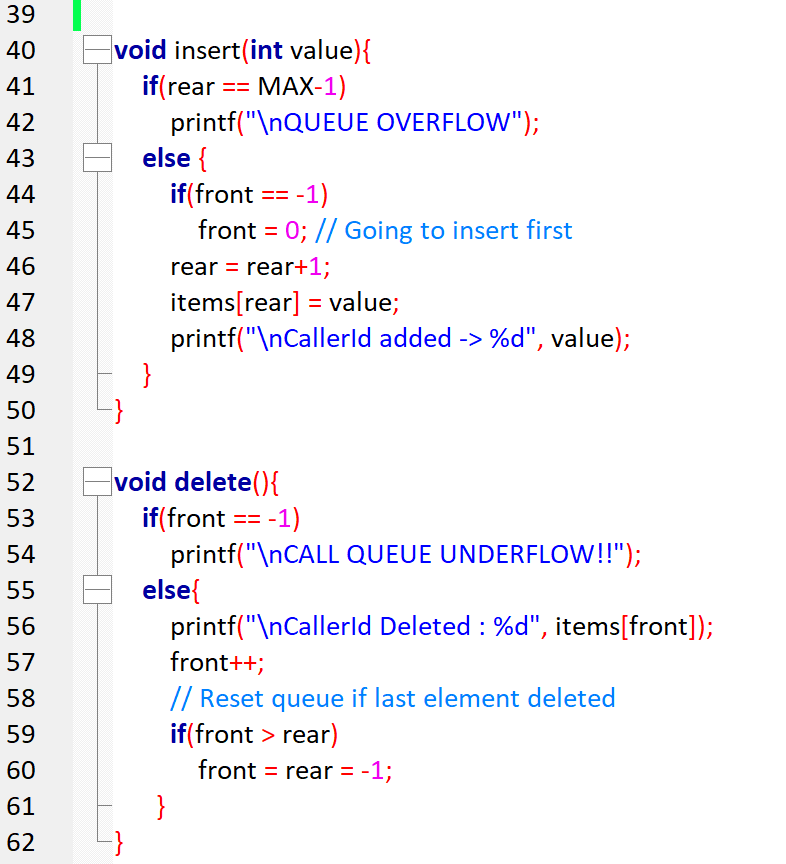
A Call center phone system has to hold the phone calls from customers and provide service based on the arrival time of the calls. Write a C program to simulate this system using appropriate data structure. Program should have options to add and remove the phone calls in appropriate order for their service.

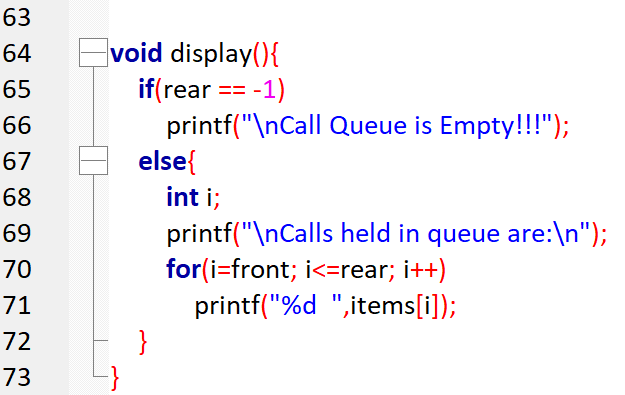
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**









****

**OUTPUT:**

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

1

Enter caller id to add

2345

CallerId added -> 2345

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

1

Enter caller id to add

7865

CallerId added -> 7865

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

1

Enter caller id to add

7777

CallerId added -> 7777

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

3

Calls held in queue are:

2345 7865 7777

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

1

Enter caller id to add

9999

QUEUE OVERFLOW

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

2

CallerId Deleted : 2345

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

3

Calls held in queue are:

7865 7777

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

2

CallerId Deleted : 7865

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

3

Calls held in queue are:

7777

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

2

CallerId Deleted : 7777

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

3

Call Queue is Empty!!!

Enter appropriate choice

1.add call

2.remove call

3.display call list

4.exit

2

CALL QUEUE UNDERFLOW!!

Enter appropriate choice

1.add call

2.remove call

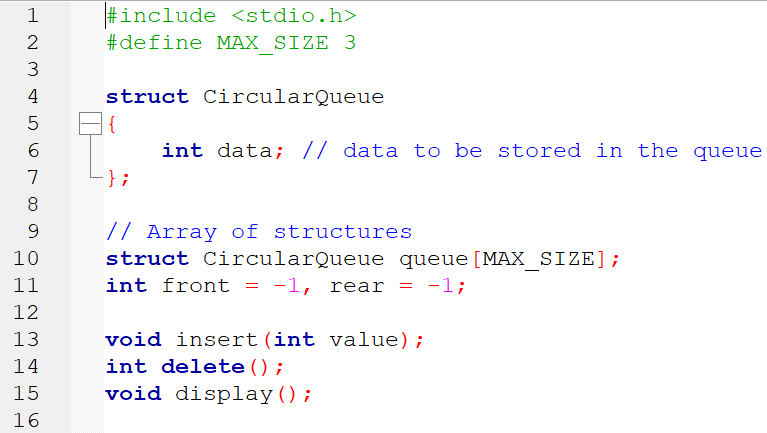
3.display call list

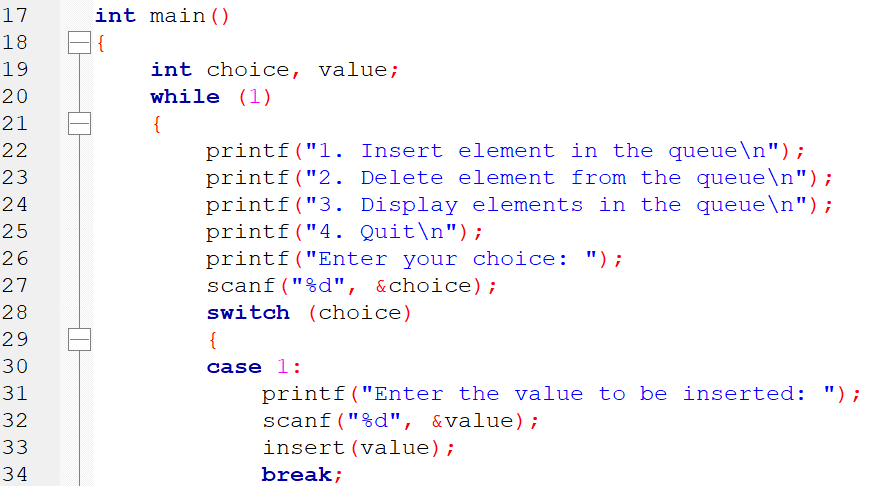
4.exit

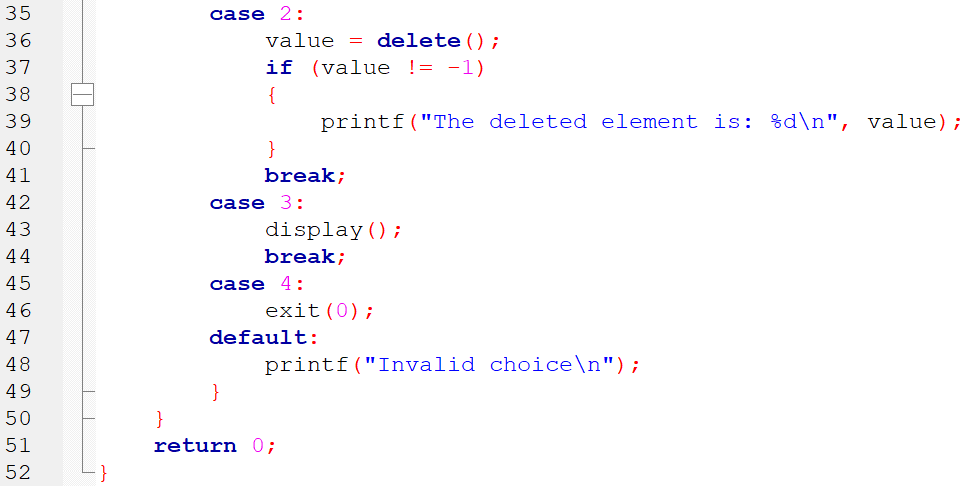
# Program 8 Circular Queue using Structure

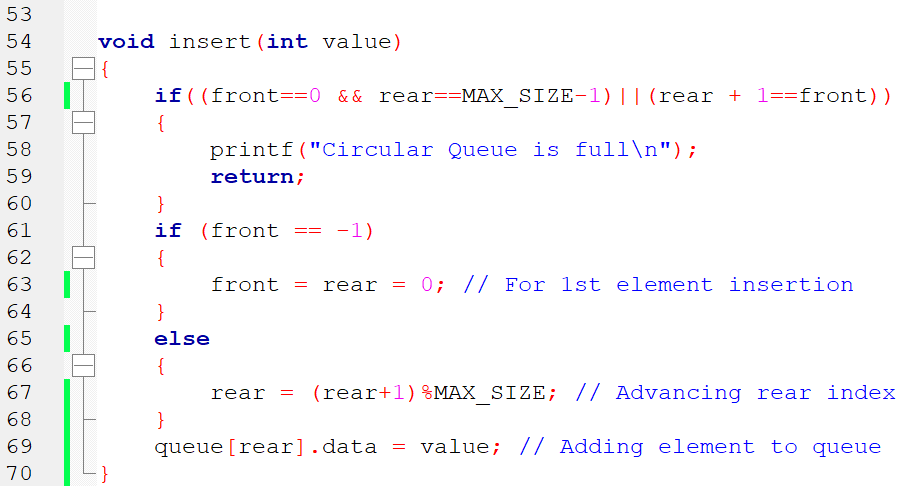
Write a C program to simulate the working of a Circular Queue of integers. Represent a circular queue element as a structure and use an array of structures as your implementation method. Start and end of the circular queue must be identified by an empty array element.

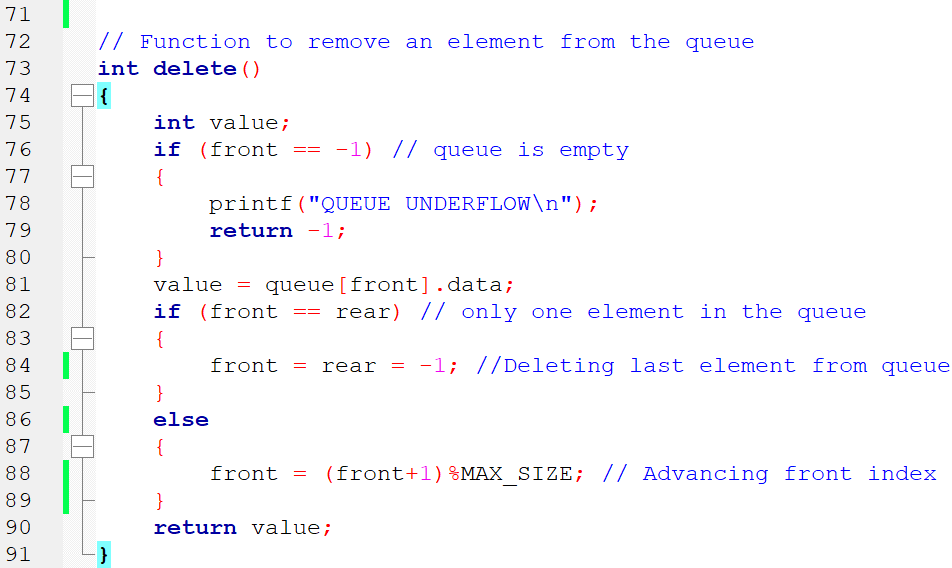
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**

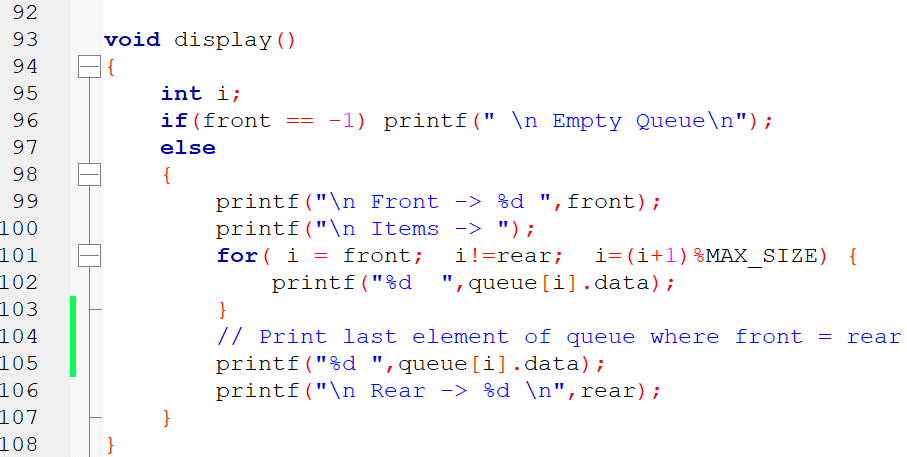












**OUTPUT**

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 1

Enter the value to be inserted: 45

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 1

Enter the value to be inserted: 67

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 3

Front -> 0

Items -> 45 67

Rear -> 1

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 1

Enter the value to be inserted: 78

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 1

Enter the value to be inserted: 99

Circular Queue is full

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 3

Front -> 0

Items -> 45 67 78

Rear -> 2

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 2

The deleted element is: 45

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 3

Front -> 1

Items -> 67 78

Rear -> 2

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 1

Enter the value to be inserted: 99

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 3

Front -> 1

Items -> 67 78 99

Rear -> 0

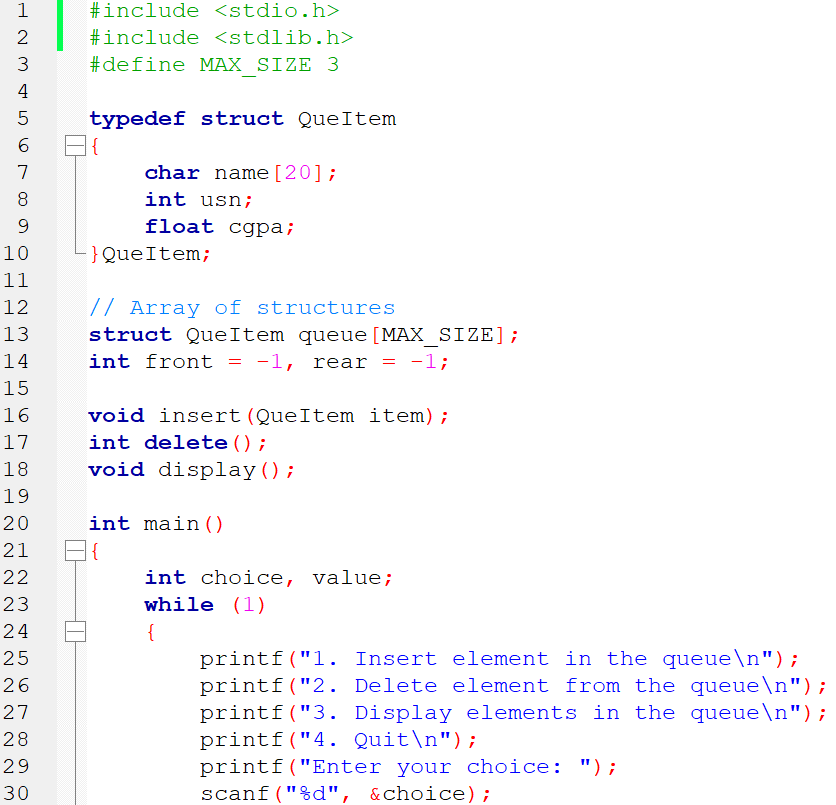
1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

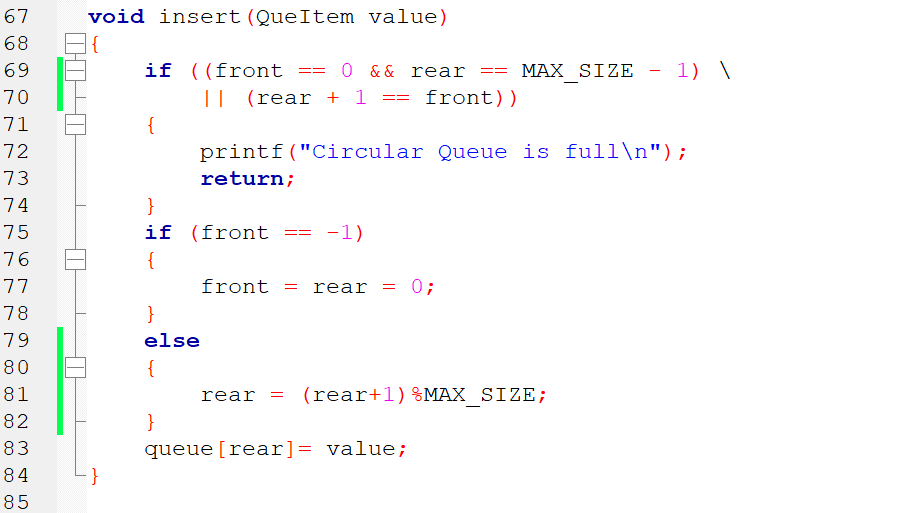
4. Quit

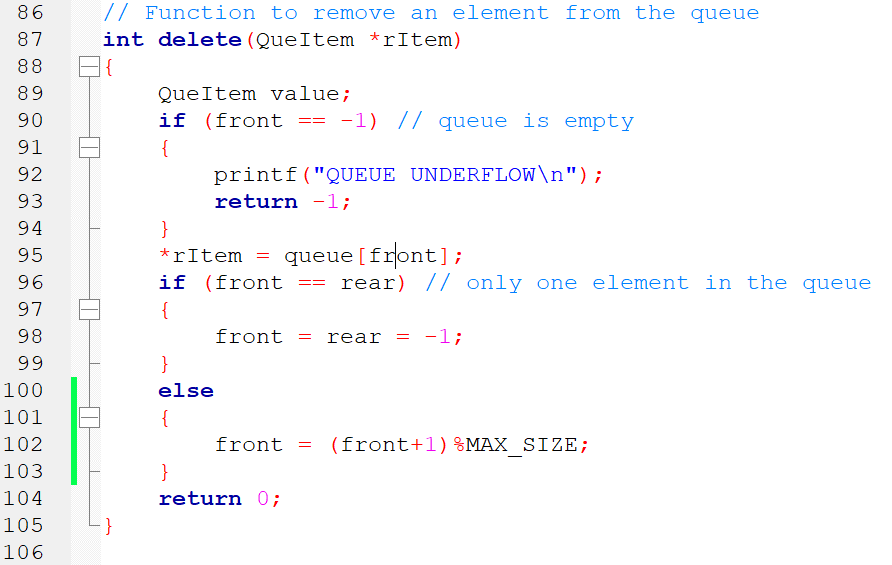
**Improved solution for CQ of student data**

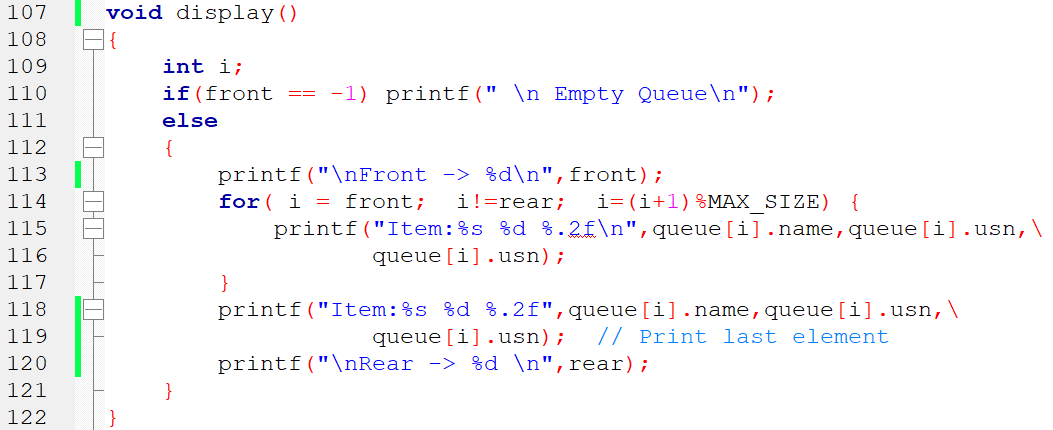












**OUTPUT:**

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 1

Enter name:amit

Enter usn:123

Enter cgpa:8.99

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 3

Front -> 0

Item:amit 123 0.00

Rear -> 0

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 1

Enter name:rahul

Enter usn:3456

Enter cgpa:9.56

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

Enter your choice: 3

Front -> 0

Item:amit 123 0.00

Item:rahul 3456 0.00

Rear -> 1

1. Insert element in the queue

2. Delete element from the queue

3. Display elements in the queue

4. Quit

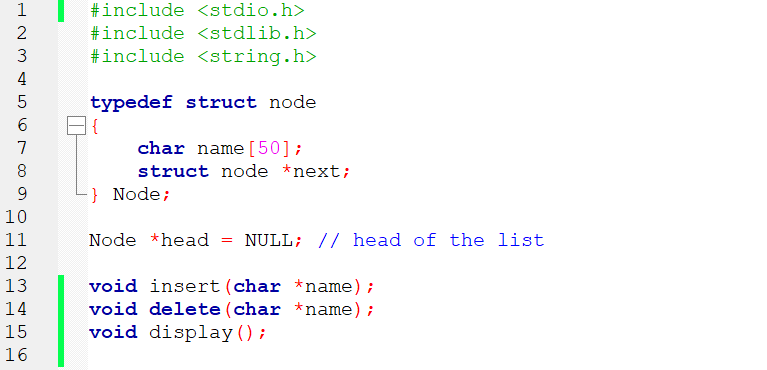
# Program 9 Linked list for sorted names

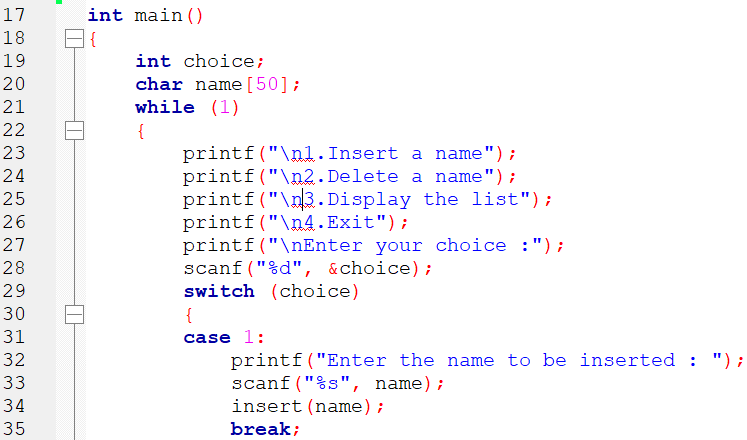
Write a program to create a singly linked list that maintains a list of names in alphabetical order. Implement the following operations on the list.

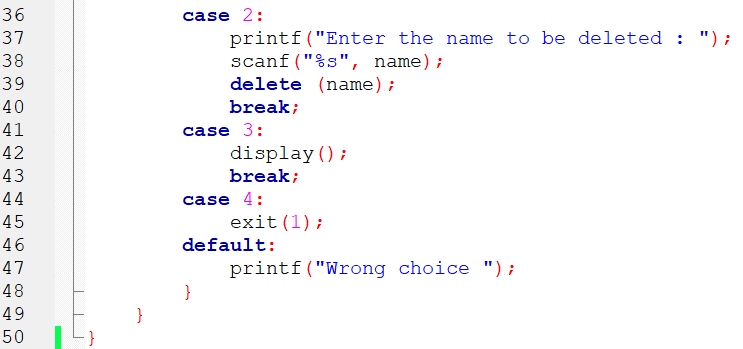
a. Insert a new name

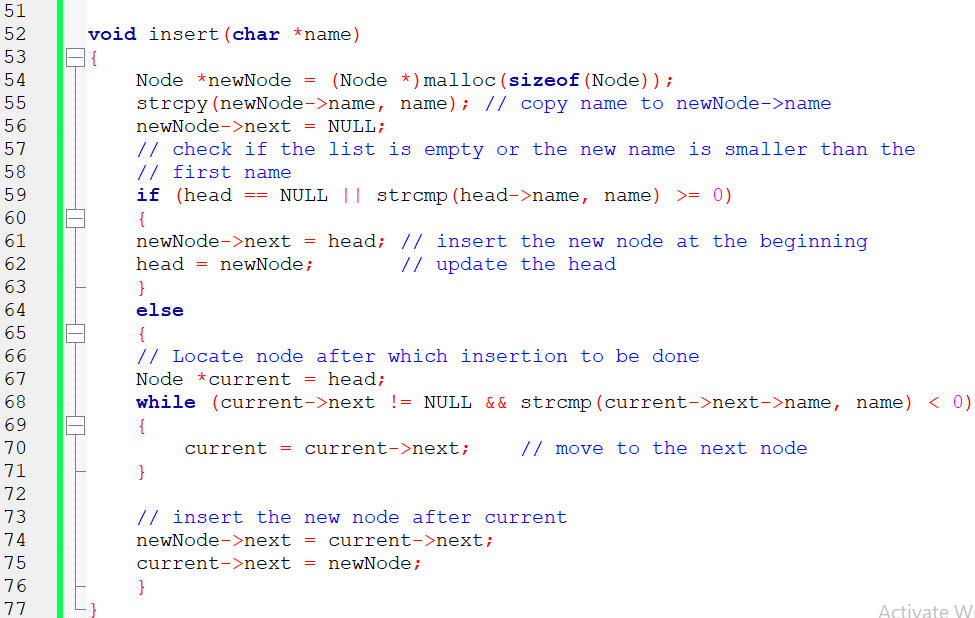
b. Delete a specified name

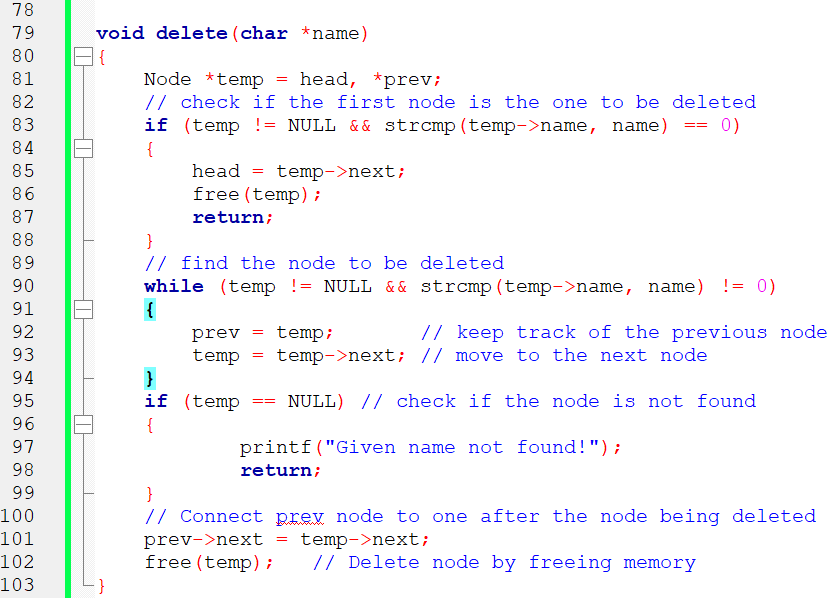
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**

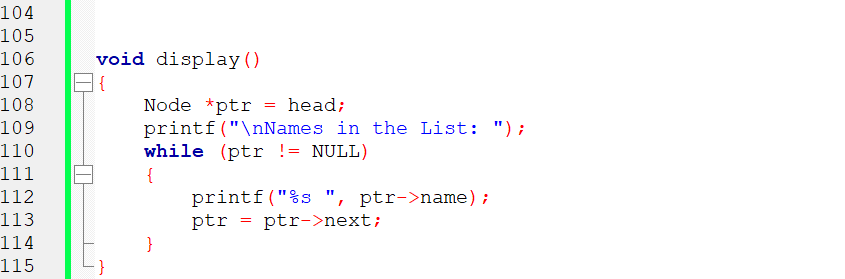
****

****

****

****

****

****

**OUTPUT:**

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :1

Enter the name to be inserted : Rohan

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :1

Enter the name to be inserted : Akshay

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :3

Names in the List: Akshay Rohan

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :1

Enter the name to be inserted : Kiran

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :3

Names in the List: Akshay Kiran Rohan

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :2

Enter the name to be deleted : Kiran

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :3

Names in the List: Akshay Rohan

1.Insert a name

2.Delete a name

3.Display the list

4.Exit

Enter your choice :2

Enter the name to be deleted : Shekar

Given name not found!

1.Insert a name

2.Delete a name

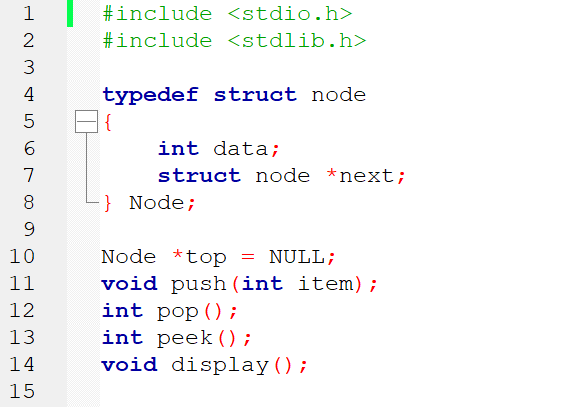
3.Display the list

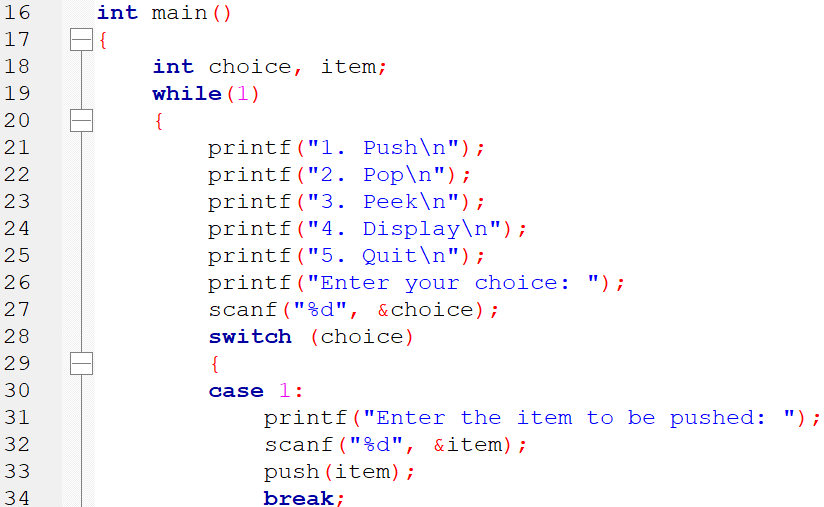
4.Exit

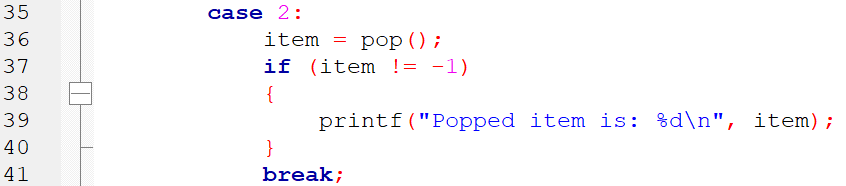
# Program 10 Stack using Linked List

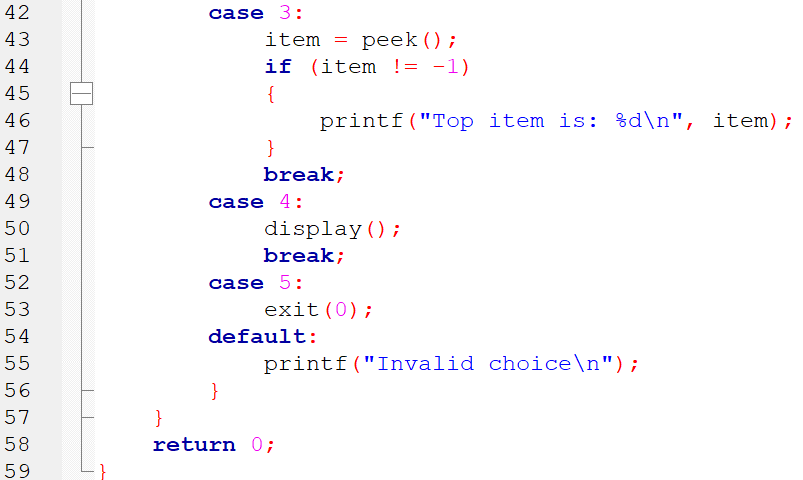
Write a C program to maintain a stack of integers using a linked list implementation method.

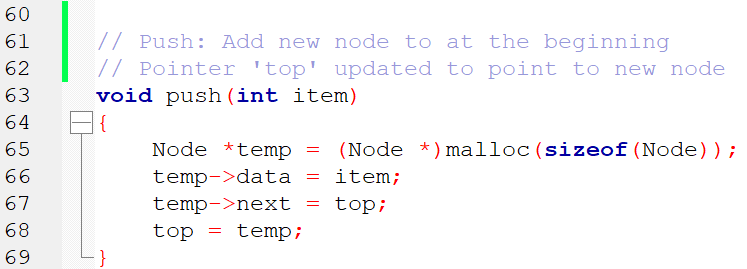
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**

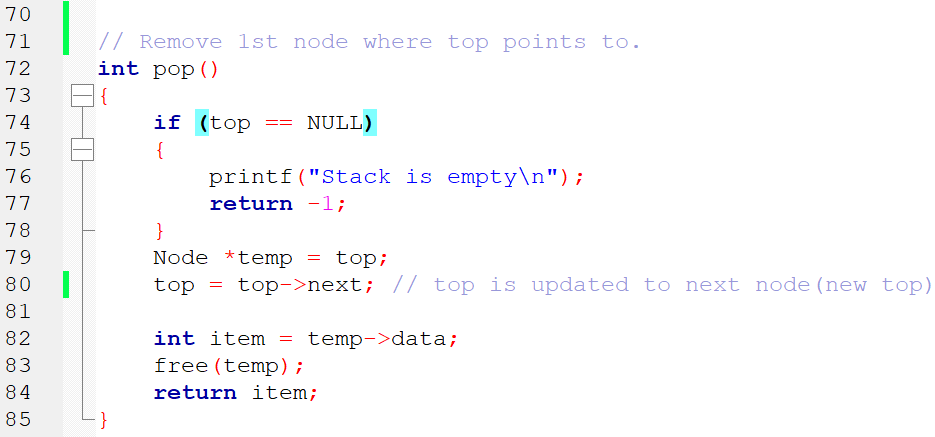


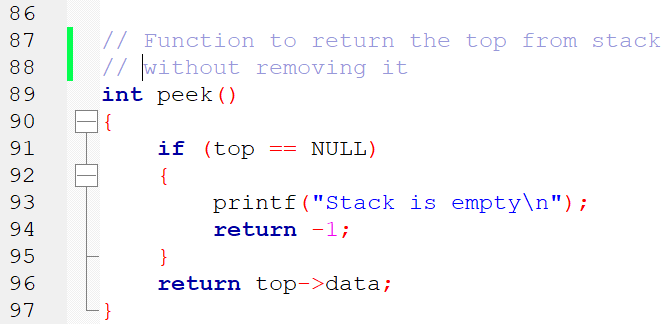


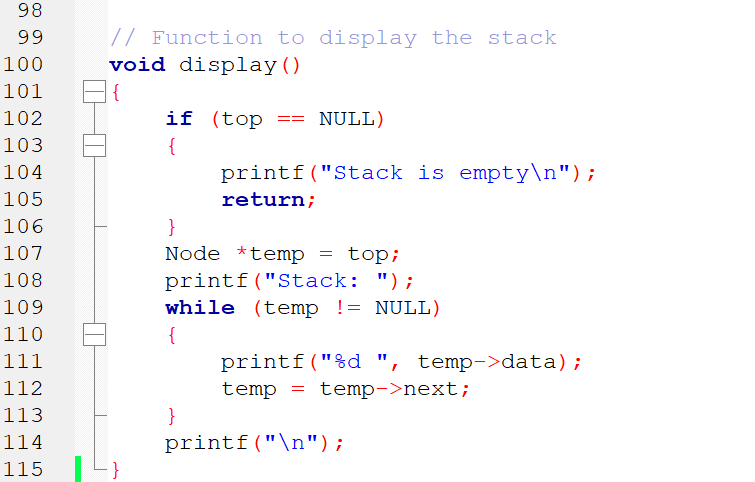


****

****

****

****

****

OUTPUT:

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 1

Enter the item to be pushed: 45

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 1

Enter the item to be pushed: 56

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 3

Top item is: 56

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 2

Popped item is: 56

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 1

Enter the item to be pushed: 99

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 4

Stack: 99 45

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 2

Popped item is: 99

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 2

Popped item is: 45

1. Push

2. Pop

3. Peek

4. Display

5. Quit

Enter your choice: 2

Stack is empty

# 

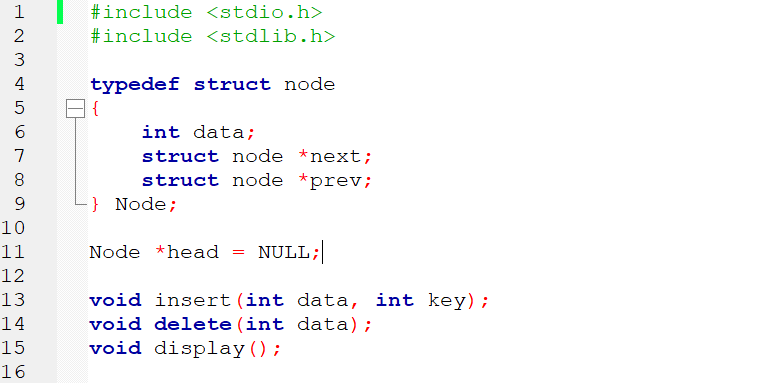
# Program 11 Doubly Linked List implementation

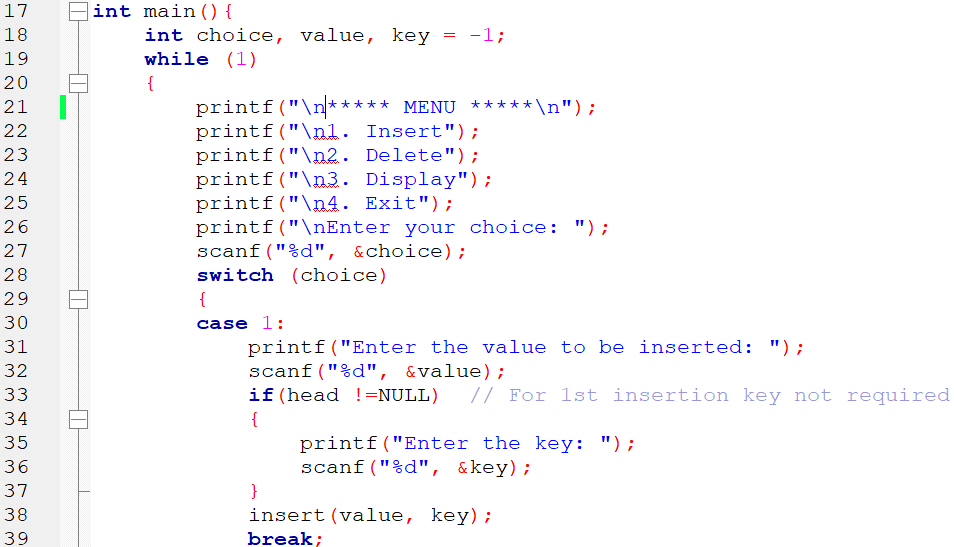
Write a C program to support the following operations on a doubly linked list.

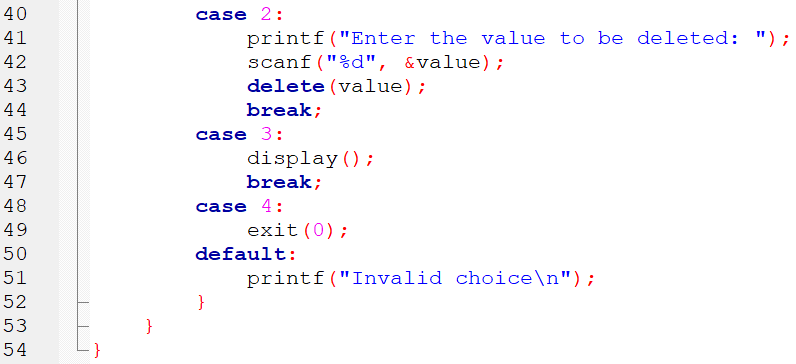
a) Insert a new node to the left of the node whose key value is read as an input.

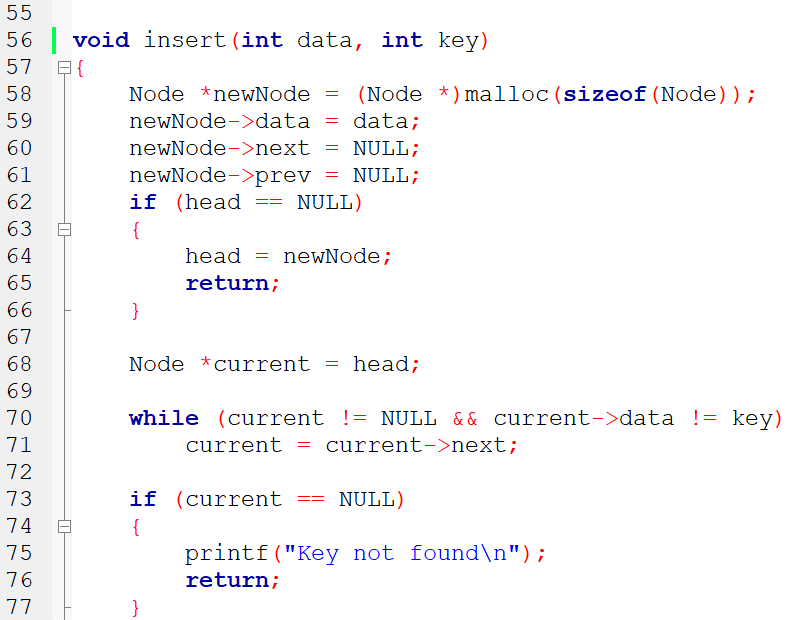
b) Delete a node with given data, if it is found otherwise display appropriate error message.

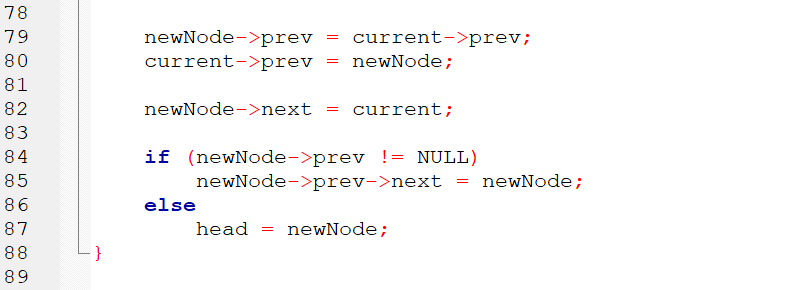
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**

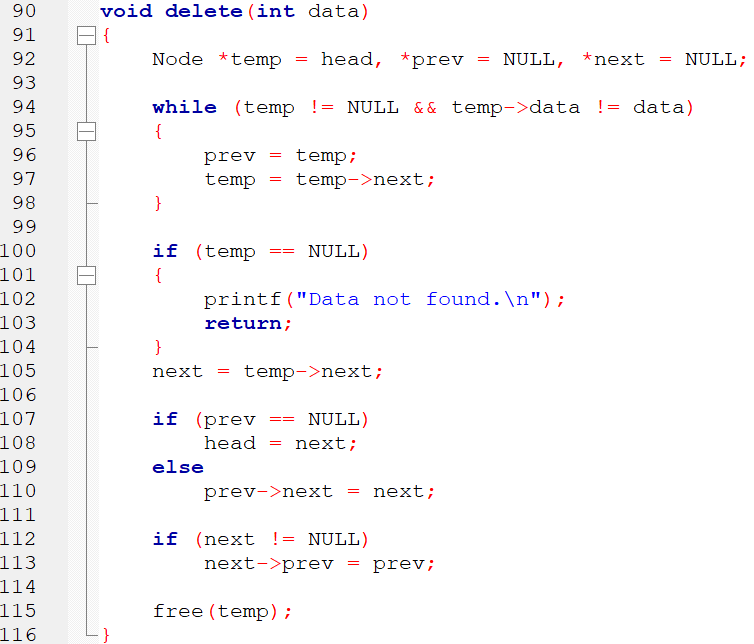


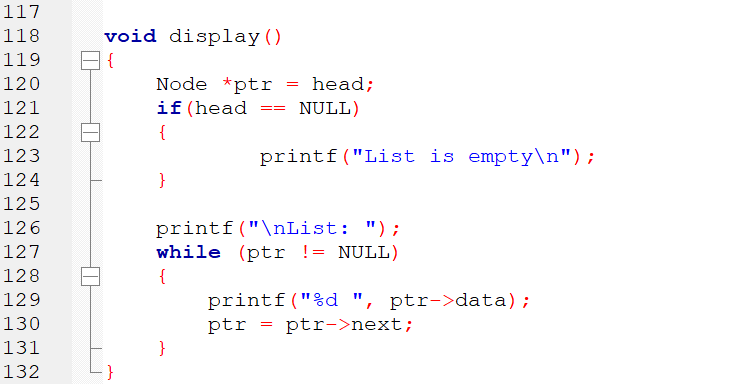


****

****

****

****

****

**OUTPUT**

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 1

Enter the value to be inserted: 50

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 1

Enter the value to be inserted: 70

Enter the key: 50

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 3

List: 70 50

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 1

Enter the value to be inserted: 60

Enter the key: 50

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 3

List: 70 60 50

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 2

Enter the value to be deleted: 60

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 3

List: 70 50

\*\*\*\*\* MENU \*\*\*\*\*

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 2

Enter the value to be deleted: 100

Data not found.

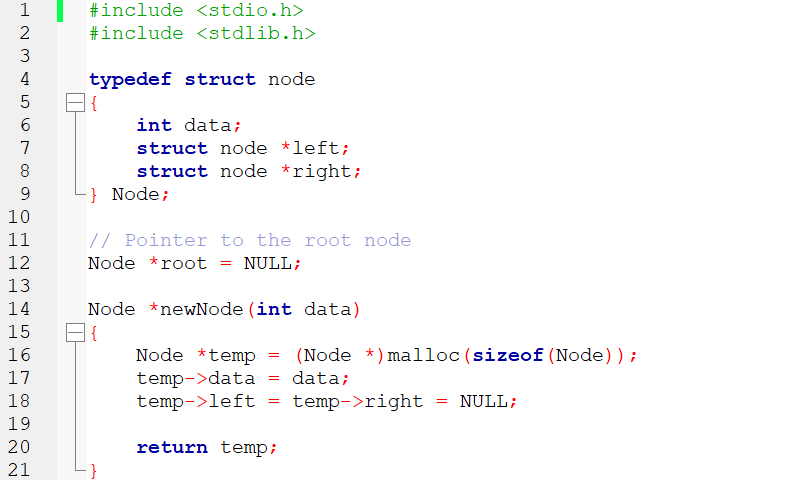
# Program 12 Binary Search Tree implementation

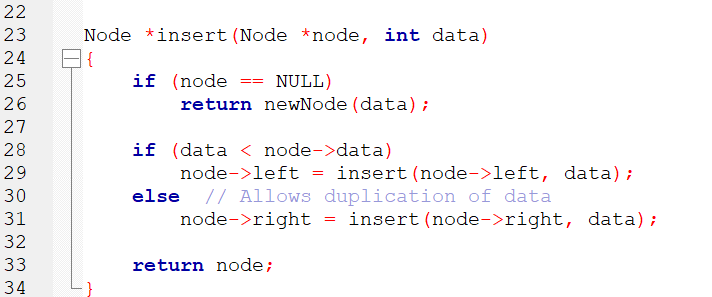
Write a C program

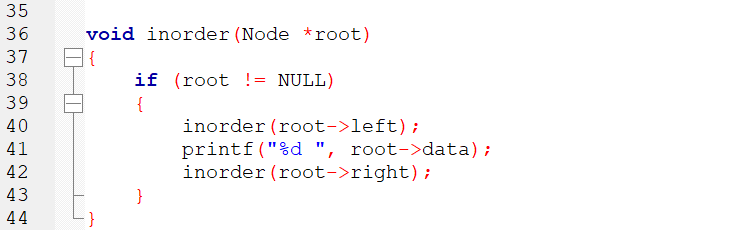
a) To construct a binary search tree of integers.

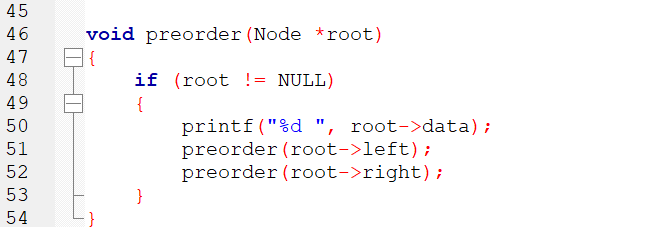
b) To traverse the tree using inorder, preorder and postorder traversal methods

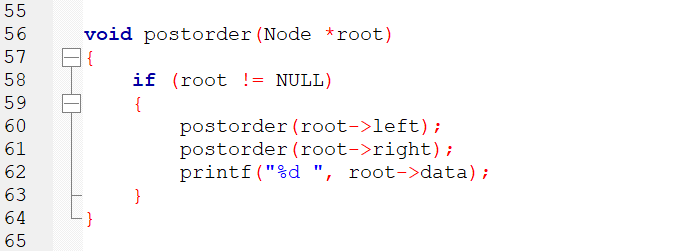
**Warning! You are warned against blindly copying the code statements. Read the code and comments to understand the implementation. Then, try to do it yourself. Blindly re-typing the code statements is a waste of time & effort.**

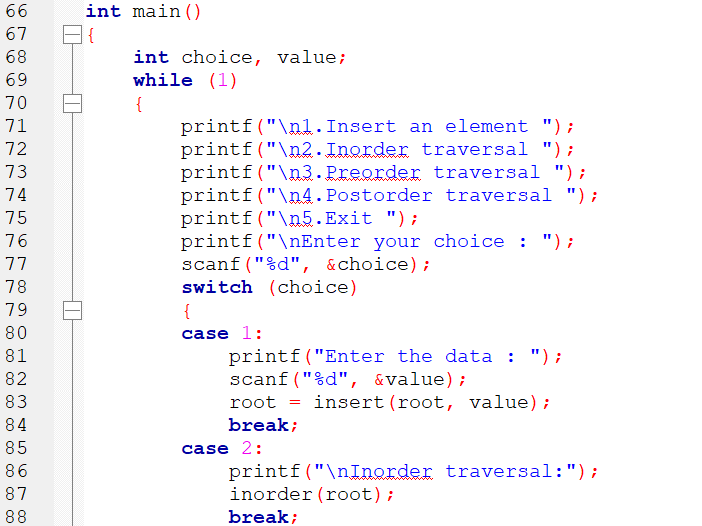


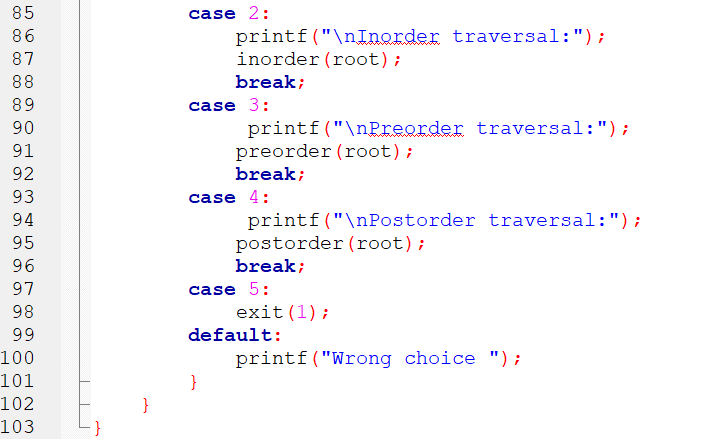












OUTPUT

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 1

Enter the data : 40

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 1

Enter the data : 20

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 1

Enter the data : 60

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 1

Enter the data : 10

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 1

Enter the data : 30

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 2

Inorder traversal:10 20 30 40 60

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 3

Preorder traversal:40 20 10 30 60

1.Insert an element

2.Inorder traversal

3.Preorder traversal

4.Postorder traversal

5.Exit

Enter your choice : 4

Postorder traversal:10 30 20 60 40