CS ASSIGNMENT

NAME: Ananya Prasad

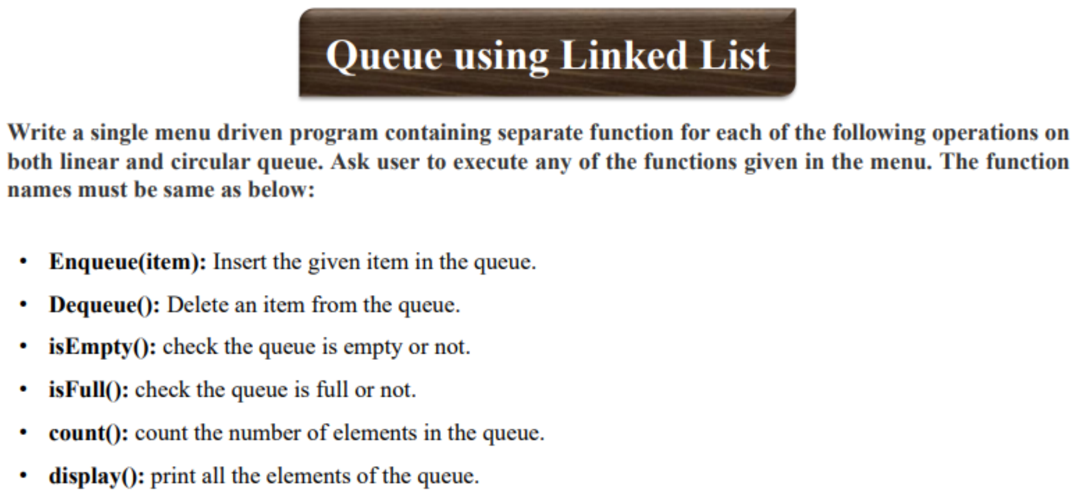
REG NO: 20BCE10093

CODE/SLOT: CSE2002/C11+DC1

FACULTY: Ms. Meenakshi Choudhary

SEMESTER: Interim Sem 2021

DATE: 25 Aug, 2021



Queue using Linked List

Write a single menu driven program containing separate function for each of the following operations on both linear and circular queue. Ask user to execute any of the functions given in the menu. The function names must be same as below:

- **Enqueue(item):** Insert the given item in the queue.
- **Dequeue():** Delete an item from the queue.
- **isEmpty():** check the queue is empty or not.
- **isFull():** check the queue is full or not.
- **count():** count the number of elements in the queue.
- **display():** print all the elements of the queue.

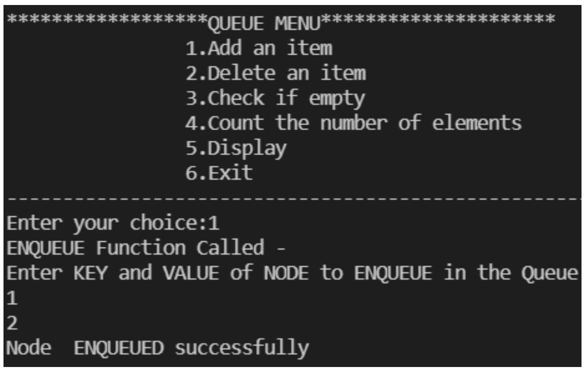
```
#include<iostream>
using namespace std;
class Node
{
  public:
  int key;
        int data; // value
        Node * next;
  Node()
  {
    key = 0;
    data = 0;
    next = NULL;
  }
  Node(int k, int d)
  {
    key = k;
    data = d;
    next = NULL;
  }
```

```
};

class Queue
{
  public:
             Node *front;
             Node *rear;

    Queue()
             {
      front = NULL;
      rear = NULL;
    }

    void isEmpty()
    {
             if(front==NULL && rear==NULL)
             {
                         cout<<"Queue is empty";
                         }
                         else
                         {
                                     cout<<"Queue is not empty";
                         }
             }



             void enqueue(Node *n)
   {
    if(front==NULL && rear==NULL)
   {
     front = n;
     rear = n;
     cout<<"Node  ENQUEUED successfully"<<endl;
   }
  else
  {
   rear->next=n;
   rear=n;
   //top = n;
   cout<<"Node  ENQUEUED successfully"<<endl;
  }

 }

        Node* dequeue()
  {
     Node *temp=NULL;
   if (front==NULL && rear==NULL)
   {
      cout << "Queue is Empty" << endl;
      return NULL;
   }
   else
   {
     if(front==rear)
     {
      temp=front;
      front = NULL;
      rear = NULL;
```

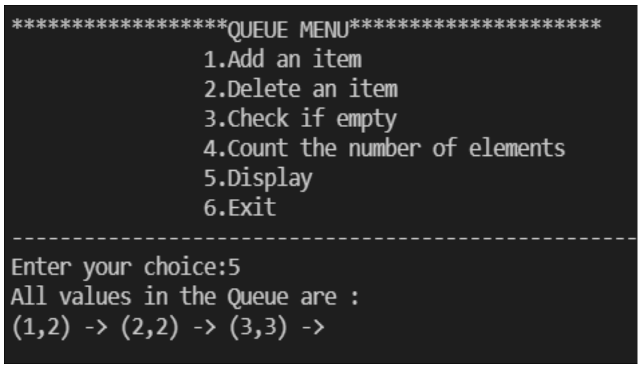
```
        return temp;
      }
      else
      {
        temp=front;
        front = front->next;
        return temp;
      }

     }
   }

            int count()
   {
     int count=0;
     Node *temp=front;
     while(temp!=NULL)
     {
       count++;
       temp=temp->next;
           }
    return count;
   }

            void display()
   {
     if(front==NULL && rear==NULL)
     {
       cout << "Queue is Empty" << endl;
     }
    else
    {
      cout << "All values in the Queue are :" << endl;
        Node *temp=front;
        while(temp!=NULL)
        {
          cout<<"("<<temp->key<<","<<temp->data<<")"<<" -> ";
          temp=temp->next;
            }
      cout<<endl;
    }

    }



};

int main()
{   Queue q;
    int option,key, data;
            int ch=0;
            int value;
            while(ch!=7)
            {
        cout<<"\n******************QUEUE MENU*********************\n";
                        cout<<"\t\t1.Add an item\n";
                        cout<<"\t\t2.Delete an item\n";
                        cout<<"\t\t3.Check if empty\n";
                        cout<<"\t\t4.Count the number of elements\n";
                        cout<<"\t\t5.Display\n";
        cout<<"\t\t6.Exit\n";
        cout<<"---------------------------------------------------\n";
        cout<<"Enter your choice:";
        cin>>ch;
Node * new_node = new Node();
        if(ch==1)
        {
              cout << "ENQUEUE Function Called -" <<endl;
  cout << "Enter KEY and VALUE of NODE to ENQUEUE in the Queue"<<endl;
  cin >> key;
  cin >> data;
  new_node->key = key;
  new_node->data = data;
  q.enqueue(new_node);
        }
        else if(ch==2)
        {
          new_node = q.dequeue();
  cout<<"Dequeued Value is: ("<<new_node->key<<","<<new_node->data<<")";
  delete new_node;
          cout<<endl;
        }
        else if(ch==3)
        {
          cout << "isEmpty Function Called - " << endl;
  q.isEmpty();
        }
        else if(ch==4)
        {
          cout << "No of nodes in the Queue: " <<q.count()<<endl;
```

```
                }
                else if(ch==5)
                {
        q.display();
        cout << endl;
                }
        }


    }
```

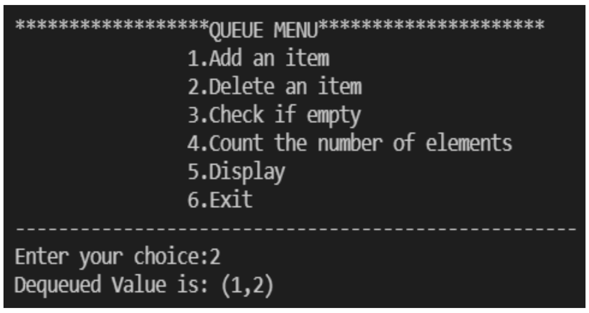
OUTPUT



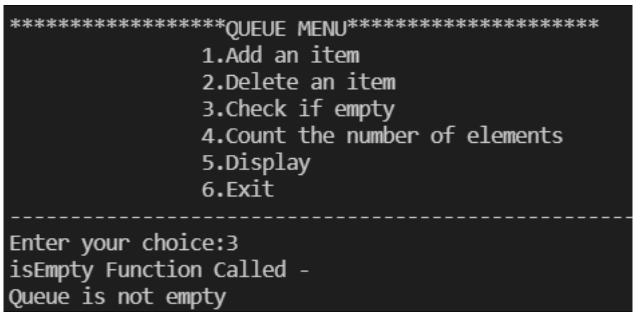
```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Count the number of elements
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:1
ENQUEUE Function Called -
Enter KEY and VALUE of NODE to ENQUEUE in the Queue
1
2
Node  ENQUEUED successfully
```



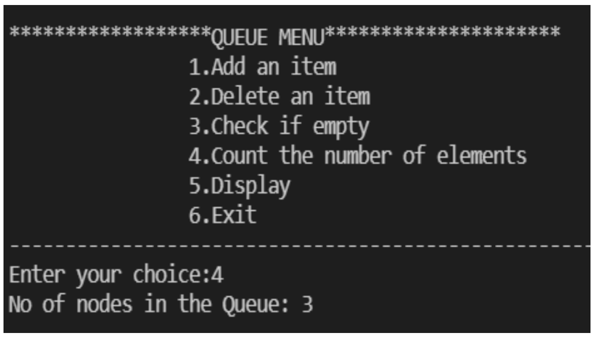
```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Count the number of elements
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:5
All values in the Queue are :
(1,2) -> (2,2) -> (3,3) ->
```



```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Count the number of elements
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:2
Dequeued Value is: (1,2)
```



```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Count the number of elements
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:3
isEmpty Function Called -
Queue is not empty
```



```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Count the number of elements
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:4
No of nodes in the Queue: 3
```

Queue using Array

Implement the stack using array and write a single menu driven program containing separate function for each of the following operations on both linear queue and circular queue. Ask user to execute any of the functions given in the menu. The function names must be same as below:

- **Enqueue(item):** Insert the given item in the queue.
- **Dequeue():** Delete an item from the queue.
- **isEmpty():** check the queue is empty or not.
- **isFull():** check the queue is full or not.
- **count():** count the number of elements in the queue.
- **display():** print all the elements of the queue.

```
#include <iostream>
#include<stdio.h>
#include<conio.h>
#include <bits/stdc++.h>
#define SIZE 15
using namespace std;

int q[SIZE];
int front = -1;
int rear = -1;
int ecount = 0;

void enqueue(int value)
{
  if(rear == SIZE-1)
  {
     "\nFull Queue. Insertion is not possible.";
  }
```

```
    else
    {
        if(front == -1)
        {
            front = 0;
        }
        rear++;
        q[rear] = value;
        cout<<"\nInsertion successful";
    }
}

void dequeue()
{
   if(front == rear)
   {
       cout<<"\nEmpty Queue, Deletion is not possible";
   }
   else
   {
       cout<<"\nDeleted:"<<q[front];
       front++;
       if(front == rear)
       {
           front = rear = -1;
       }

   }
}
```

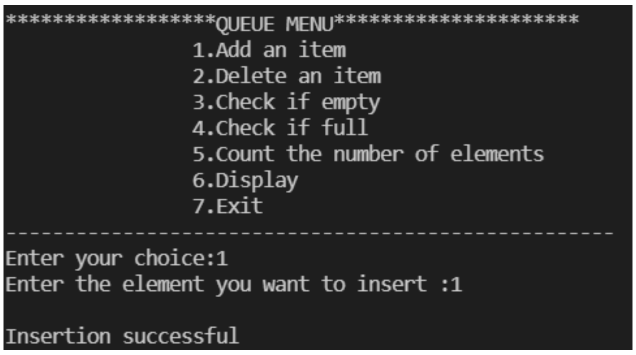
```
void isempty()
{
   if(front < 0 || front > rear)
   {
       cout<<"Queue is empty";
   }
   else
   {
       cout<<"Queue is not empty";
   }
}
void isfull()
{
   if(rear == SIZE - 1)
   {
       cout<<"\nQueue is full\n";
   }
   else
   {
       cout<<"\nQueue is not full";
   }
}
void count()
{
    int i;
      for(i=front; i<=rear; i++)
      {
          ecount++;
      }
    cout<<"The number of elements in the queue : "<<ecount;
```

```
}
void display()
{
  if(rear == -1)
  {
     cout<<"\nEmpty Queue";
  }
  else
  {
    int i;
    cout<<"\nQueue elements :\n";
    for(i=front; i<=rear; i++)
    {
       cout<<q[i];
    }
  }
}

int main()
{
        int ch=0;
        int value;
        while(ch!=7)
        {       cout<<"\n******************QUEUE MENU*********************\n";
                cout<<"\t\t1.Add an item\n";
                cout<<"\t\t2.Delete an item\n";
                cout<<"\t\t3.Check if empty\n";
                cout<<"\t\t4.Check if full\n";
                cout<<"\t\t5.Count the number of elements\n";
                cout<<"\t\t6.Display\n";
```

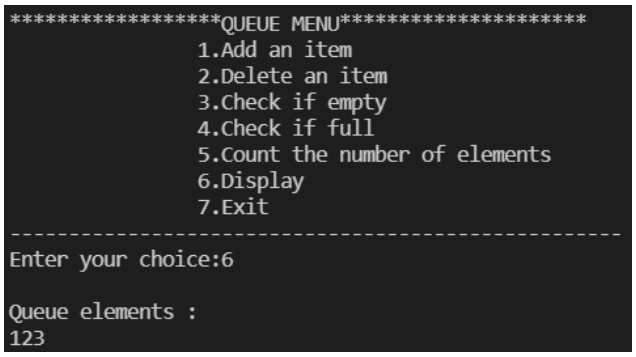
```
cout<<"\t\t7.Exit\n";
cout<<"---------------------------------------------------\n";
cout<<"Enter your choice:";
cin>>ch;
if(ch==1)
{
   cout<<"Enter the element you want to insert :";
   cin>>value;
         enqueue(value);
}
else if(ch==2)
{
   dequeue();
}
else if(ch==3)
{
   isempty();
}
else if(ch==4)
{
   isfull();
}
else if(ch==5)
{
   count();
}
else if(ch==6)
{
   display();
}
```

```
        }


    }
```

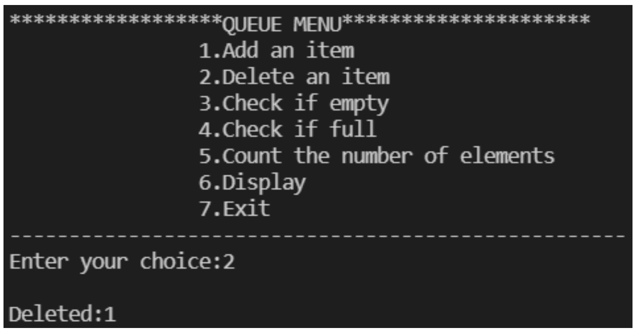
OUTPUT



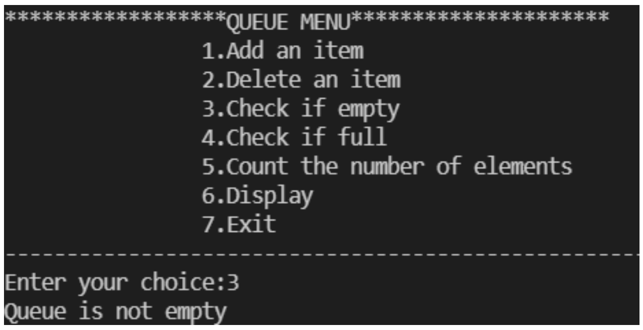
```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Count the number of elements
                6.Display
                7.Exit
---------------------------------------------------
Enter your choice:1
Enter the element you want to insert :1

Insertion successful
```



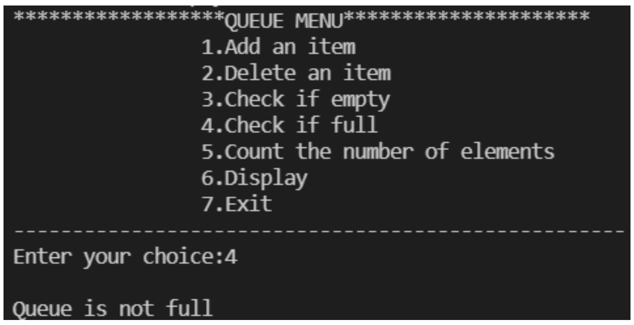
```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Count the number of elements
                6.Display
                7.Exit
---------------------------------------------------
Enter your choice:6

Queue elements :
123
```



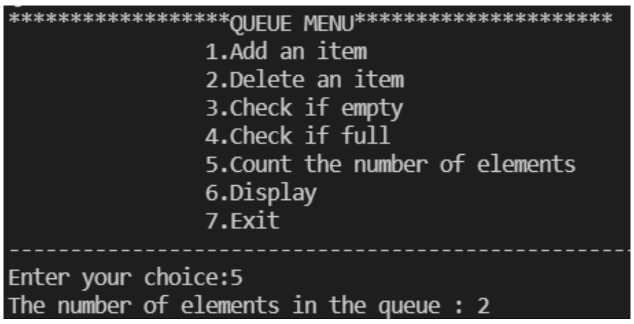
```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Count the number of elements
                6.Display
                7.Exit
---------------------------------------------------
Enter your choice:2

Deleted:1
```



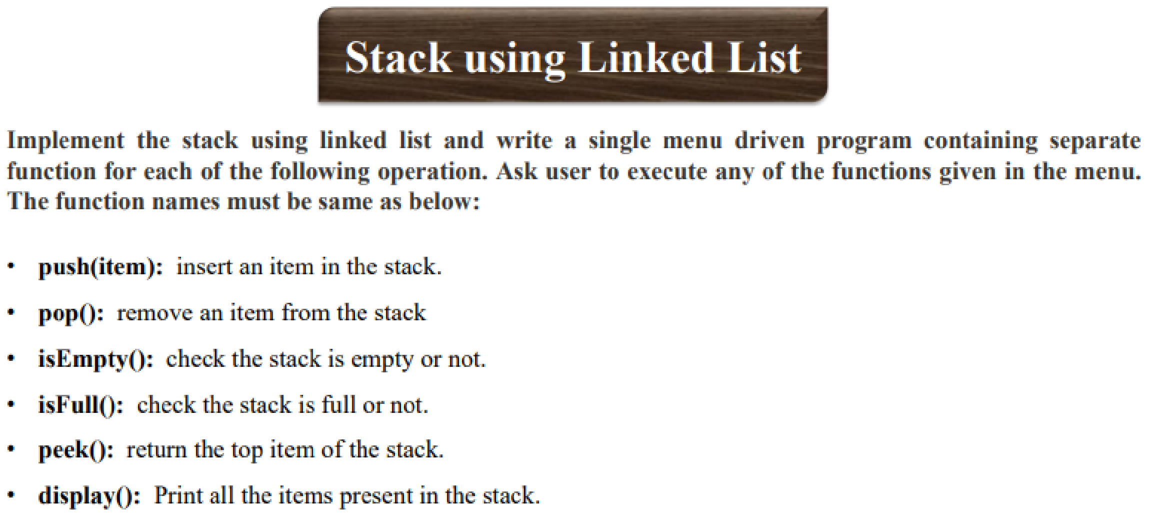
```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Count the number of elements
                6.Display
                7.Exit
---------------------------------------------------
Enter your choice:3
Queue is not empty
```



```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Count the number of elements
                6.Display
                7.Exit
---------------------------------------------------
Enter your choice:4

Queue is not full
```



```
******************QUEUE MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Count the number of elements
                6.Display
                7.Exit
---------------------------------------------------
Enter your choice:5
The number of elements in the queue : 2
```



Stack using Linked List

Implement the stack using linked list and write a single menu driven program containing separate function for each of the following operation. Ask user to execute any of the functions given in the menu. The function names must be same as below:

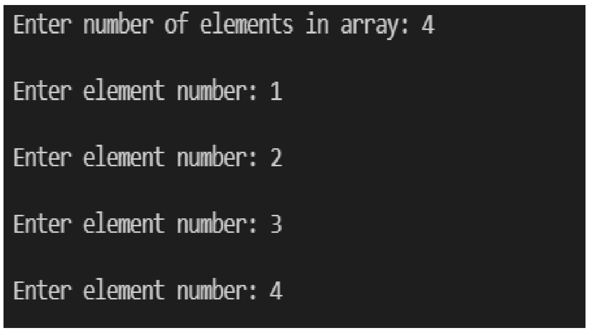
- **push(item):** insert an item in the stack.
- **pop():** remove an item from the stack
- **isEmpty():** check the stack is empty or not.
- **isFull():** check the stack is full or not.
- **peek():** return the top item of the stack.
- **display():** Print all the items present in the stack.

```
#include <iostream>

using namespace std;


class Node

{

public:

  Node *next;

  int data;

};


void push(Node **head_ref, int item)

{

  Node *new_node = new Node();

  new_node->data = item;

  if (*head_ref == NULL)

  {

    new_node->next = NULL;

    *head_ref = new_node;
  }
  else
  {
    new_node->next = *head_ref;
    *head_ref = new_node;
  }
  return;
}

void pop(Node **head_ref)
{
  Node *tmp = *head_ref;
  *head_ref = tmp->next;
  tmp->next = NULL;
}

void isEmpty(Node **head_ref)
{
  if (*head_ref == NULL)
  {
        cout<<"Stack is empty";
  }
  else
  {
      cout<<"Stack is not empty.";
  }
}

void isFull() {}   //cannot define this as we don’t know the when the stack ends in linked list
implementation.
```

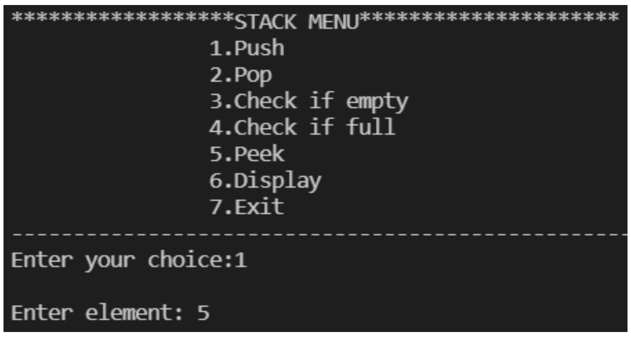
```
int peek(Node *head)
{
  return head->data;
}

void display(Node *head)
{
  Node *tmp = head;
  while (tmp->next != NULL)
  {
    cout << " " << tmp->data;
    tmp = tmp->next;
  }
  cout << " " << tmp->data;
}

int main()
{
    int ch=0;
        int item;
    int size;
    cout << "\nEnter number of elements in array: ";
    cin >> size;
    Node *head = NULL;
    for (int i = 0; i < size; i++)
    {
        int x;
        cout << "\nEnter element number: ";
        cin >> x;
        push(&head, x);
    }


      while(ch!=7)
      {         cout<<"\n******************STACK MENU*********************\n";
                cout<<"\t\t1.Push\n";
                cout<<"\t\t2.Pop\n";
                cout<<"\t\t3.Check if empty\n";
                cout<<"\t\t4.Check if full\n";
                cout<<"\t\t5.Peek\n";
                cout<<"\t\t6.Display\n";
                cout<<"\t\t7.Exit\n";
                cout<<"--------------------------------------------------\n";
                cout<<"Enter your choice:";
                cin>>ch;
                if(ch==1)
                {
                   cout << "\nEnter element: ";
          cin >> item;
          push(&head, item);
                }
                else if(ch==2)
                {
                   pop(&head);
                }
                else if(ch==3)
                {
                   isEmpty(&head);
                }
                else if(ch==4)
                {
```

```
            isFull();
        }
        else if(ch==5)
        {
            peek(head);
        }
        else if(ch==6)
        {
            display(head);
        }
    }
}
```

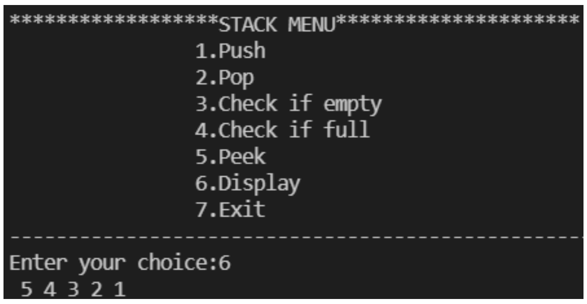
OUTPUT



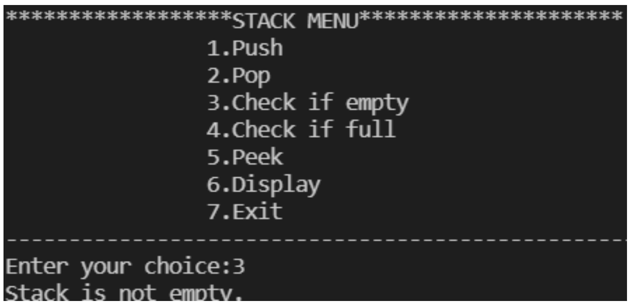
```
Enter number of elements in array: 4

Enter element number: 1

Enter element number: 2

Enter element number: 3

Enter element number: 4
```



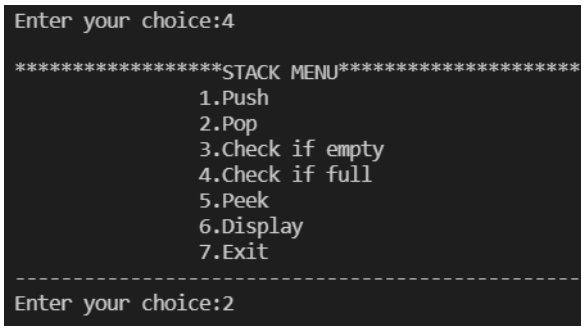
```
*****************STACK MENU*********************
                1.Push
                2.Pop
                3.Check if empty
                4.Check if full
                5.Peek
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:1

Enter element: 5
```



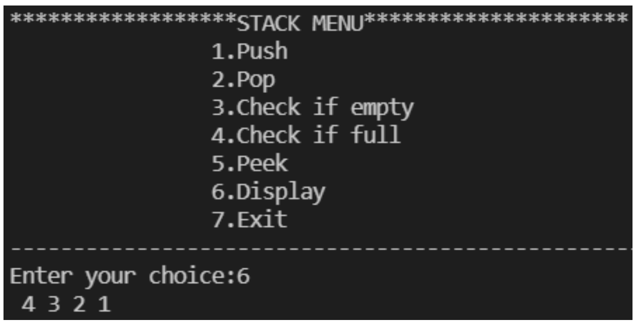
```
*****************STACK MENU*********************
                1.Push
                2.Pop
                3.Check if empty
                4.Check if full
                5.Peek
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:6
 5 4 3 2 1
```



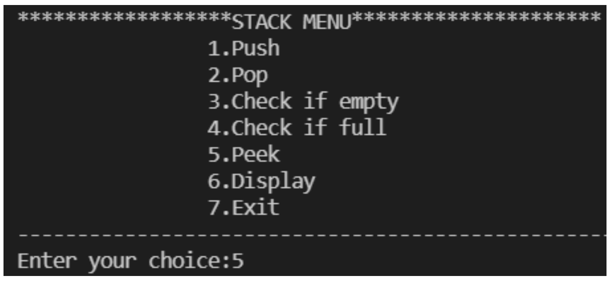
```
*****************STACK MENU*********************
                1.Push
                2.Pop
                3.Check if empty
                4.Check if full
                5.Peek
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:3
Stack is not empty.
```



```
Enter your choice:4

*****************STACK MENU*********************
                1.Push
                2.Pop
                3.Check if empty
                4.Check if full
                5.Peek
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:2
```



```
*****************STACK MENU*********************
                1.Push
                2.Pop
                3.Check if empty
                4.Check if full
                5.Peek
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:6
 4 3 2 1
```



```
*****************STACK MENU********************
                1.Push
                2.Pop
                3.Check if empty
                4.Check if full
                5.Peek
                6.Display
                7.Exit
--------------------------------------------------
Enter your choice:5
```

Stack using Array

Implement the stack using array data structure and write a single menu driven program containing separate function for each of the following operation. Ask user to execute any of the functions given in the menu. The function names must be same as below:

- **push(item):** insert an item in the stack.
- **pop():** remove an item from the stack
- **isEmpty():** check the stack is empty or not.
- **isFull():** check the stack is full or not.
- **peek():** return the top item of the stack.
- **display():** Print all the items present in the stack.

```
#include <iostream>

using namespace std;

#define MAX 1000


class Stack

{

private:

  int top;


public:

  int stack_array[MAX];


  Stack()

  {
```

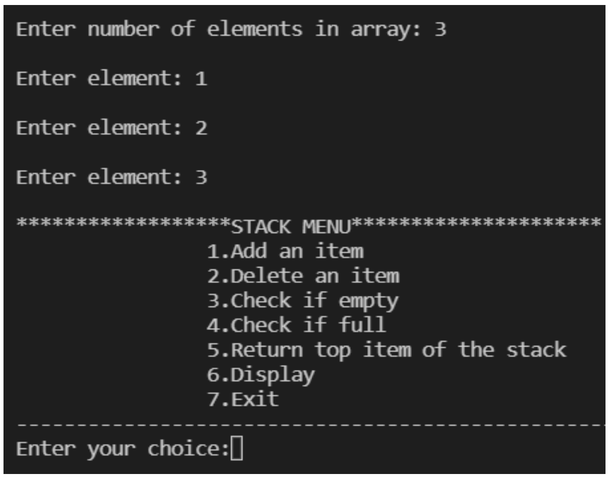
```
  top = -1;
}

void push(int item)
  {
      if (top >= (MAX - 1))
      {
          cout << "\nStack Overflow!";
          return;
      }
      else
      {
          top = top + 1;
          stack_array[top] = item;
          return;
      }
  }
void isEmpty()
{
      if (top == -1)
      {
          cout<<"Stack is empty";
      }
      else
          cout<<"Stack is not empty.";
  }
void isFull()
{
      if (top >= (MAX - 1))
      {
            cout<<"Stack is full.";
        }
        else
            cout<<"Stack is not full.";
        }
void peek()
 {
        if (top != -1)
        {
            cout<<stack_array[top];
        }
        else
            cout<<"stack is empty";
 }
void display()
 {
        if (top == -1)
        {
            cout << ""\nstack is empty";
            return;
        }
        else
        {
            for (int i = top; i >= 0; i--)
            {
                cout << stack_array[i] << " ";
            }
        }
  }
void pop()
```

```
   {
       if (top == -1)
       {
           cout << "\nStack is empty";
           return;
       }
       else
       {
           top = top - 1;
           cout << "\nelement deleted.";
       }

   }
};

int main()
{
       int ch=0;
    int size;
    cout << "\nEnter number of elements in array: ";
    cin >> size;
    class Stack s;
    for (int i = 0; i < size; i++)
    {
        int x;
        cout << "\nEnter element: ";
        cin >> x;
        s.push(x);
    }
       int item;
```

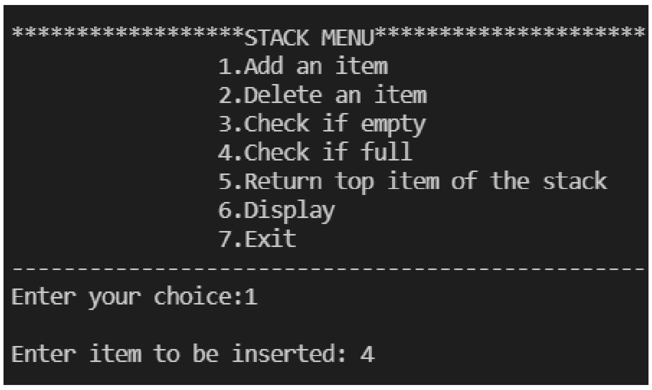
```
while(ch!=7)
{         cout<<"\n*******************STACK MENU**********************\n";
          cout<<"\t\t1.Add an item\n";
          cout<<"\t\t2.Delete an item\n";
          cout<<"\t\t3.Check if empty\n";
          cout<<"\t\t4.Check if full\n";
          cout<<"\t\t5.Return top item of the stack\n";
          cout<<"\t\t6.Display\n";
          cout<<"\t\t7.Exit\n";
          cout<<"--------------------------------------------------\n";
          cout<<"Enter your choice:";
          cin>>ch;
          if(ch==1)
          {   cout << "\nEnter item to be inserted: ";
              cin >> item;
    s.push(item);
          }
          else if(ch==2)
          {
    s.pop();
          }
          else if(ch==3)
          {
              s.isEmpty();
          }
          else if(ch==4)
          {
              s.isFull();
          }
          else if(ch==5)
```

```
                {
                   s.peek();
                }
                else if(ch==6)
                {
                   s.display();
                }
        }


}
```

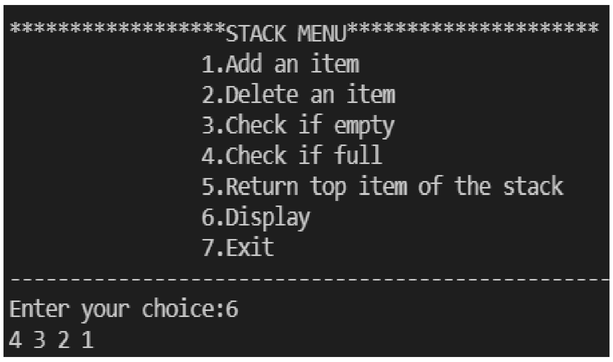
OUTPUT



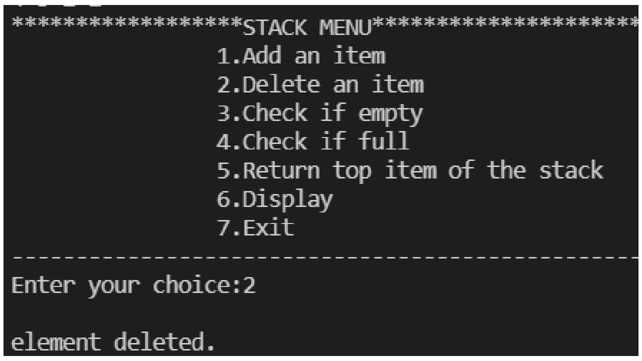
```
Enter number of elements in array: 3

Enter element: 1

Enter element: 2

Enter element: 3

******************STACK MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Return top item of the stack
                6.Display
                7.Exit
-------------------------------------------------
Enter your choice:
```



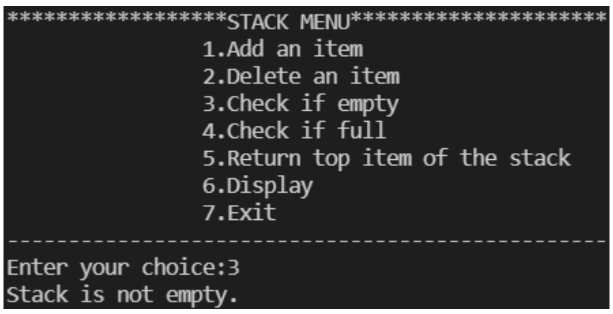
```
******************STACK MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Return top item of the stack
                6.Display
                7.Exit
-------------------------------------------------
Enter your choice:1

Enter item to be inserted: 4
```



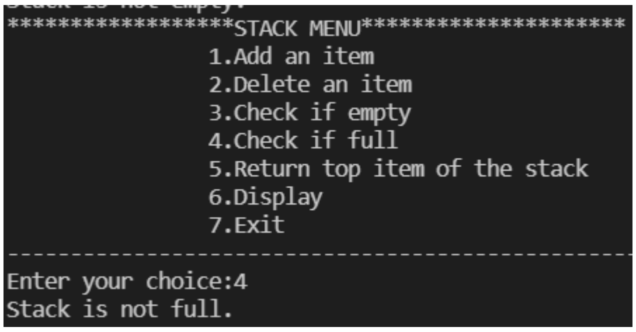
```
******************STACK MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Return top item of the stack
                6.Display
                7.Exit
-------------------------------------------------
Enter your choice:6
4 3 2 1
```



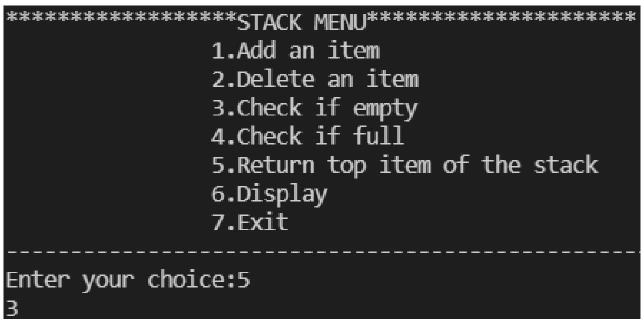
```
******************STACK MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Return top item of the stack
                6.Display
                7.Exit
-------------------------------------------------
Enter your choice:2

element deleted.
```



```
*****************STACK MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Return top item of the stack
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:3
Stack is not empty.
```



```
*****************STACK MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Return top item of the stack
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:4
Stack is not full.
```



```
*****************STACK MENU*********************
                1.Add an item
                2.Delete an item
                3.Check if empty
                4.Check if full
                5.Return top item of the stack
                6.Display
                7.Exit
------------------------------------------------
Enter your choice:5
3
```

Doubly Linked List

Write a single menu driven program containing separate function for each of the following operation on doubly linear Linked List. Ask user to execute functions of choice given in the menu. The function names must be same as below:

- **Insert_begin():** insert a new node at the beginning of the list.
- **Insert_last():** insert a new node at the last in the list.
- **Insert_random(pos):** insert a new node at the given position in the list.
- **Insert_specific(key):** insert a new node after the node containing given key.
- **Delete_begin():** delete node from beginning of the list.
- **Delete_end():** delete the last node from the list.
- **Search(key):** search the given key in the list and return the node's position containing the key.
- **Display():** print all the elements of the list.

```
#include <iostream>
using namespace std;

class Node
{
public:
  int data;
  Node *prev;
  Node *next;
};

void delete_Atlast(Node **head_ref)
{
  if (*head_ref == NULL)
  {
    cout << "\n List is empty";
    return;
  }
  Node *tmp = *head_ref;
  while (tmp->next != NULL)
  {
    tmp = tmp->next;
  }
  tmp->prev->next = NULL;
  tmp->prev = NULL;
}

void delete_Atfirst(Node **head_ref)
{
  Node *tmp = *head_ref;
  tmp->next->prev = NULL;
  *head_ref = tmp->next;
  tmp->next = NULL;
}

void display(Node *head)
{
  Node *tmp = head;
  if (tmp != NULL)
  {
    while (tmp->next != NULL)
    {
      cout << tmp->data << " ";
      tmp = tmp->next;
    }
    cout << tmp->data << " ";
  }
}

void insert_Atfirst(Node **head_ref, int item)
{
  Node *new_node = new Node();
  new_node->prev = NULL;
  new_node->data = item;
  new_node->next = *head_ref;
  *head_ref = new_node;
}

void insert_random(Node *head, int pos, int item)
{
  Node *tmp = head;
  int count = 1;
  while (count < pos - 1)
  {
    count++;
    tmp = tmp->next;
  }
  Node *new_node = new Node();
  new_node->next = tmp->next;
  tmp->next->prev = new_node;
  new_node->prev = tmp;
  tmp->next = new_node;
  new_node->data = item;
}

void insert_Atlast(Node **head_ref, int item)
{
  Node *new_node = new Node();
  new_node->data = item;
  new_node->prev = new_node->next = NULL;
  if (*head_ref == NULL)
  {
    *head_ref = new_node;
  }
  else
  {
    Node *tmp = *head_ref;
    while (tmp->next != NULL)
    {
      tmp = tmp->next;
    }
    tmp->next = new_node;
    new_node->prev = tmp;
  }
}

void search(Node *head)
```

```
{
   Node *ptr;
   int item,i=0,flag;
   ptr = head;
   if(ptr == NULL)
   {
      printf("\nEmpty List\n");
   }
   else
   {
      printf("\nEnter item which you want to search?\n");
      scanf("%d",&item);
      while (ptr!=NULL)
      {
         if(ptr->data == item)
         {
            printf("\nitem found at location %d ",i+1);
            flag=0;
            break;
         }
         else
         {
            flag=1;
         }
         i++;
         ptr = ptr -> next;
      }
      if(flag==1)
      {
         printf("\nItem not found\n");
```

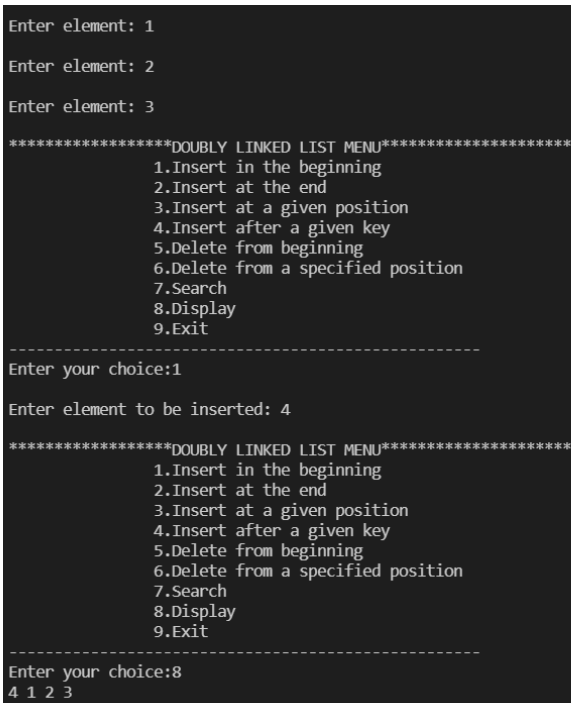
```
        }
    }

}

void keyinsert(Node *head, int key,int item)
{
  Node *tmp = head;
  while (tmp != NULL)
  {
    if (tmp->data == key)
    {
      Node *new_node = new Node();
      new_node->data = item;
      tmp->next->prev = new_node;
      new_node->next = tmp->next;
      new_node->prev = tmp;
      tmp->next = new_node;
      return;
    }
    tmp = tmp->next;
  }
  cout << "Key not found\n";
}
int main()
{
    Node *head = NULL;
    int ch = 0;
    int size;
    cout << "\nEnter number of elemnts in array: ";
```

```
cin >> size;

for (int i = 0; i < size; i++)
{
   int x;
   cout << "\nEnter element: ";
   cin >> x;
   insert_Atlast(&head, x);
}

int item, pos;
while(ch!=9)
   {        cout<<"\n*****************LINKED LIST MENU********************\n";
            cout<<"\t\t1.Insert in the beginning\n";
            cout<<"\t\t2.Insert at the end\n";
            cout<<"\t\t3.Insert at a given position\n";
            cout<<"\t\t4.Insert after a given key\n";
            cout<<"\t\t5.Delete from beginning\n";
            cout<<"\t\t6.Delete from a specified position\n";
            cout<<"\t\t7.Search\n";
            cout<<"\t\t8.Display\n";
            cout<<"\t\t9.Exit\n";
            cout<<"-------------------------------------------------\n";
            cout<<"Enter your choice:";
            cin>>ch;
            if(ch==1)
            {   cout << "\nEnter element to be inserted: ";
      int item;
      cin >> item;
      insert_Atlast(&head, item);
```

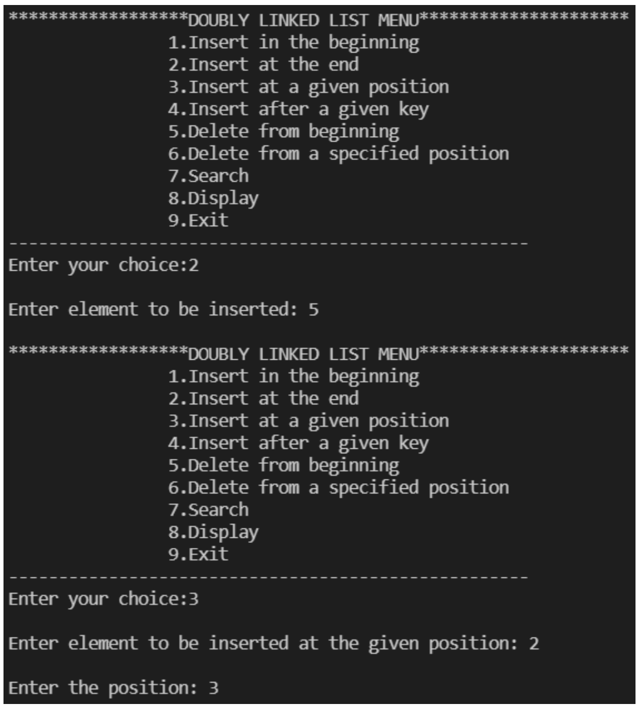
```
        }
        else if(ch==2)
        {   int item;
            cout << "\nEnter element to be inserted: ";
cin >> item;
insert_Atfirst(&head, item);
        }
        else if(ch==3)
        {   int item;
            cout << "\nEnter element to be inserted at the given position: ";
cin >> item;
int pos;
cout << "\nEnter the position: ";
cin>>pos;
insert_random(head, pos, item);
        }
        else if(ch==4)
        {   int key;
            cout<<"Enter the value you want to add.";
            cin>>item;
            cout<<"Enter the location:";
            cin>>key;
            keyinsert(head, key, item);
        }
        else if(ch==5)
        {
            delete_Atfirst(&head);
        }
        else if(ch==6)
        {
```

```
                    delete_Atlast(&head);
                }
                else if(ch==7)
                {
                    int item;
                    search(head);
                }
                else if(ch==8)
                {
                    display(head);
                }
        }
}
```

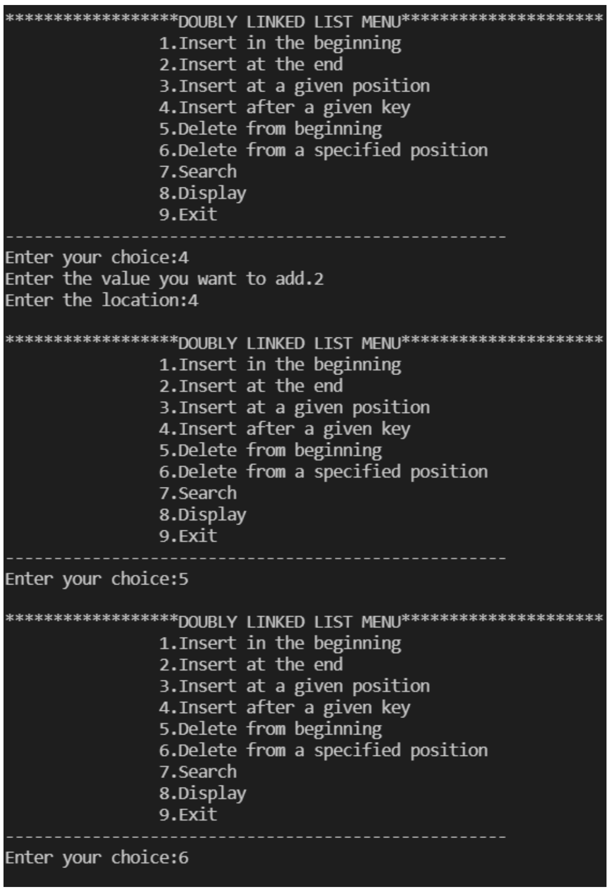
OUTPUT



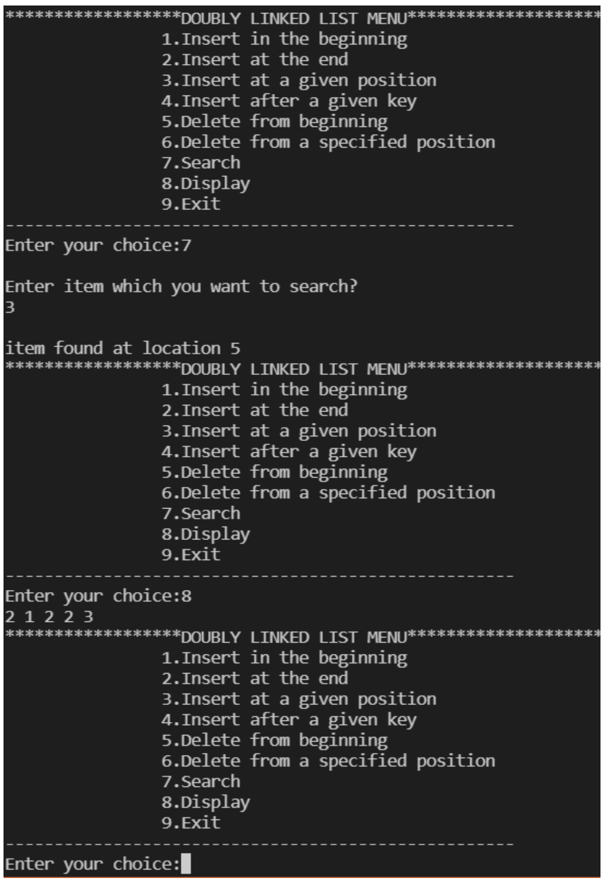
```
Enter element: 1

Enter element: 2

Enter element: 3

*****************DOUBLY LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:1

Enter element to be inserted: 4

*****************DOUBLY LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:8
4 1 2 3
```



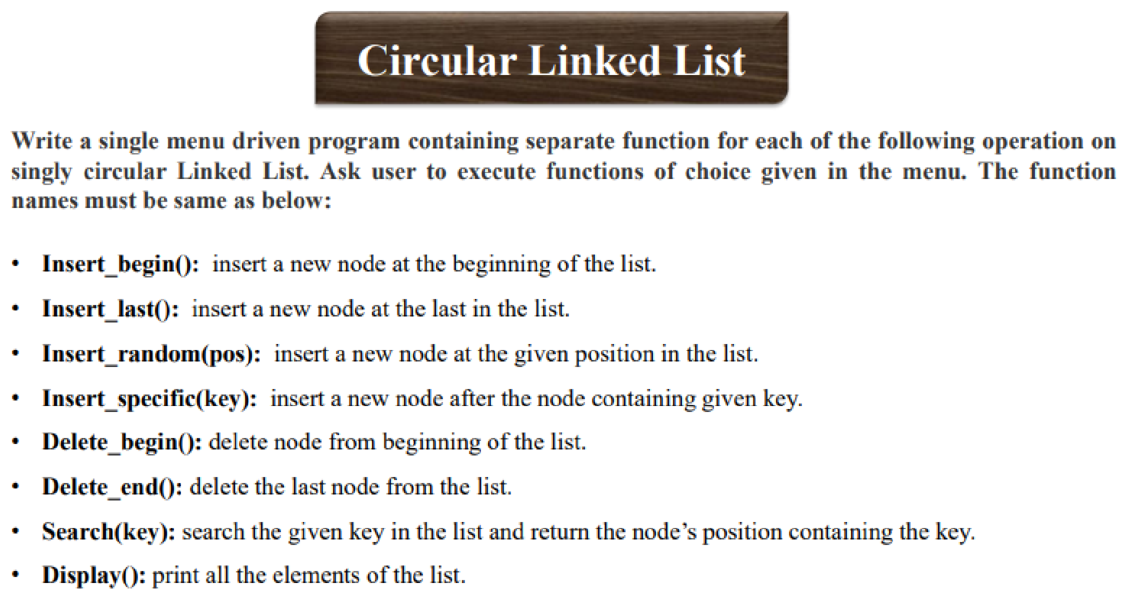
```
*****************DOUBLY LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:2

Enter element to be inserted: 5

*****************DOUBLY LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:3

Enter element to be inserted at the given position: 2

Enter the position: 3
```



```
******************DOUBLY LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:4
Enter the value you want to add.2
Enter the location:4

******************DOUBLY LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:5

******************DOUBLY LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:6
```



```
******************DOUBLY LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:7

Enter item which you want to search?
3

item found at location 5
******************DOUBLY LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:8
2 1 2 2 3
******************DOUBLY LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from a specified position
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:
```



Circular Linked List

Write a single menu driven program containing separate function for each of the following operation on singly circular Linked List. Ask user to execute functions of choice given in the menu. The function names must be same as below:

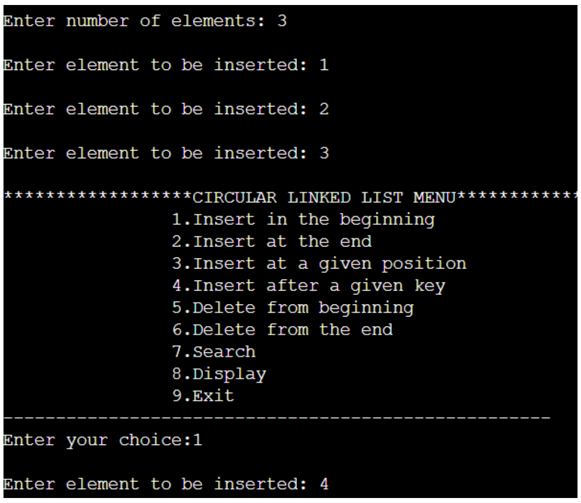
- **Insert_begin():** insert a new node at the beginning of the list.
- **Insert_last():** insert a new node at the last in the list.
- **Insert_random(pos):** insert a new node at the given position in the list.
- **Insert_specific(key):** insert a new node after the node containing given key.
- **Delete_begin():** delete node from beginning of the list.
- **Delete_end():** delete the last node from the list.
- **Search(key):** search the given key in the list and return the node's position containing the key.
- **Display():** print all the elements of the list.

```
#include <iostream>
using namespace std;

class Node
{
public:
  int data;
  Node *next;
};

void insert_Atlast(Node **head_ref, int item)
{
  Node *new_node = new Node();
  new_node->data = item;
  if (*head_ref == NULL)
  {
    *head_ref = new_node;
    new_node->next = new_node;
  }
  else
  {
    Node *tmp = *head_ref;
    while (tmp->next != *head_ref)
    {
      tmp = tmp->next;
    }
    new_node->next = *head_ref;
    tmp->next = new_node;
  }
  return;
}

void display(Node *head)
{
  Node *tmp = head;
  if (head != NULL)
  {
    do
    {
      cout << tmp->data << " ";
      tmp = tmp->next;
    } while (tmp != head);
  }
}

void insert_Atfirst(Node **head_ref, int item)
{

  Node *new_node = new Node();
  new_node->data = item;
  if (*head_ref == NULL)
  {
    *head_ref = new_node;
    new_node->next = new_node;
  }
  else
  {
    Node *tmp = *head_ref;
    while (tmp->next != *head_ref)
    {
      tmp = tmp->next;
    }
    new_node->next = *head_ref;
    tmp->next = new_node;
  }
  *head_ref = new_node;
  return;
}

void insert_random(Node *head, int pos, int item)
{
  Node *tmp = head;
  int count = 1;
  while (count < pos - 1)
  {
    count++;
    tmp = tmp->next;
  }
  Node *new_node = new Node();
  new_node->data = item;
  new_node->next = tmp->next;
  tmp->next = new_node;
  return;
}

void search(Node *head, int item)
{   Node *tmp;
    int i=0,flag=1;
    tmp = head;
    if(tmp == NULL)
    {
        cout<<"\nEmpty List\n";
    }
    else
    {
        cout<<"\nEnter item which you want to search?\n";
        cin>>item;
        if(head ->data == item)
        {
        cout<<"item found at location"<<i+1;
        flag=0;
        return;
        }
        else
        {
        while (tmp->next != head)
        {
            if(tmp->data == item)
            {
                cout<<"item found at location"<<i+1;
                flag=0;
                return;
            }
            else
            {
                flag=1;
            }
            i++;
            tmp = tmp -> next;
        }
        }
        if(flag != 0)
        {
            cout<<"Item not found\n";
            return;
        }
    }

}
void delete_Atfirst(Node **head_ref)
{
  Node *tmp = *head_ref;
  Node *last = *head_ref;
  while (last->next != *head_ref)
  {
    last = last->next;
  }
  *head_ref = tmp->next;
  tmp->next = NULL;
  last->next = *head_ref;
  return;
}

void delete_Atlast(Node **head_ref)
{
  Node *tail = *head_ref;
  Node *last = NULL;
  while (tail->next->next != *head_ref)
  {
    tail = tail->next;
  }
  last = tail->next;
  tail->next = *head_ref;
  last->next = NULL;
  return;
}
void keyinsert(int value, int location)
{
   Node *newNode;
   Node *head;
   newNode -> data = value;
   if(head == NULL)
   {
      head = newNode;
      newNode -> next = head;
   }
   else
   {
      struct Node *temp = head;
      while(temp -> data != location)
      {
         if(temp -> next == head)
         {
            cout<<"Given node is not found in the list!!!";
         }
         else
         {
            temp = temp -> next;
         }
      }
      newNode -> next = temp -> next;
      temp -> next = newNode;
      cout<<"\nInsertion success!!!";
   }
}

int main()
{
  int ch = 0;
  Node *head = NULL;
  int size;
  cout << "\nEnter number of elements: ";
  cin >> size;

  for (int i = 0; i < size; i++)
  {
    int x;
    cout<<"\nEnter element to be inserted: ";
    cin >> x;
    insert_Atlast(&head, x);
  }
  Node *temp = NULL;
       while(ch!=9)
       {          cout<<"\n******************CIRCULAR LINKED LIST
MENU*********************\n";
                  cout<<"\t\t1.Insert in the beginning\n";
                  cout<<"\t\t2.Insert at the end\n";
                  cout<<"\t\t3.Insert at a given position\n";
                  cout<<"\t\t4.Insert after a given key\n";
                  cout<<"\t\t5.Delete from beginning\n";
                  cout<<"\t\t6.Delete from a specified position\n";
                  cout<<"\t\t7.Search\n";
                  cout<<"\t\t8.Display\n";
                  cout<<"\t\t9.Exit\n";
                  cout<<"---------------------------------------------------\n";
                  cout<<"Enter your choice:";
                  cin>>ch;
                  if(ch==1)
                  {   cout << "\nEnter element to be inserted: ";
          int item;
          cin >> item;
          insert_Atfirst(&head, item);
                  }
                  else if(ch==2)
                  {   int item;
                      cout << "\nEnter element to be inserted: ";
          cin >> item;
          insert_Atleft(&head, item);
                  }
                  else if(ch==3)
```

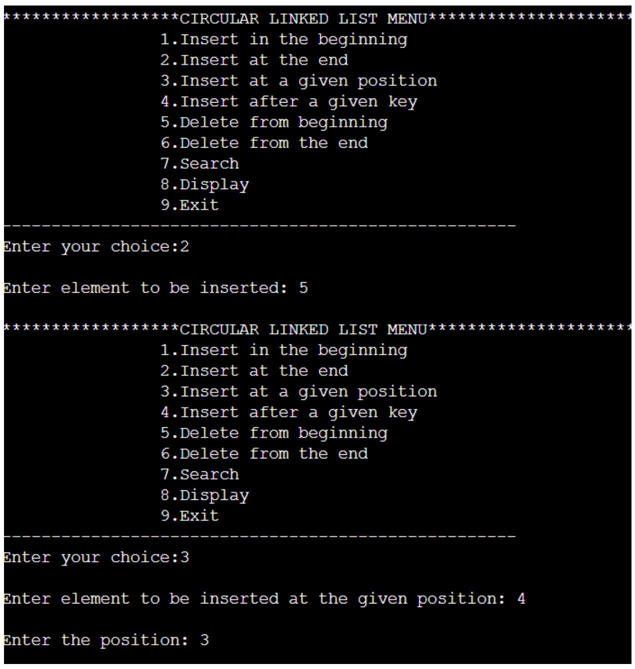
```
        {   int item;
            cout << "\nEnter element to be inserted at the given position: ";
cin >> item;
int pos;
cout << "\nEnter the position: ";
cin>>pos;
insert_random(head, pos, item);
        }
        else if(ch==4)
        {   int value, location;
            cout<<"Enter the value you want to add.";
            cin>>value;
            cout<<"Enter the location:";
            cin>>location;
            keyinsert(value, location);
        }
        else if(ch==5)
        {
            delete_Atfirst(&head);
        }
        else if(ch==6)
        {
            delete_Atlast(&head);
        }
        else if(ch==7)
        {
            int item;
            search(head,item);
        }
        else if(ch==8)
```

```
                {
                    display(head);
                }
        }

}
```

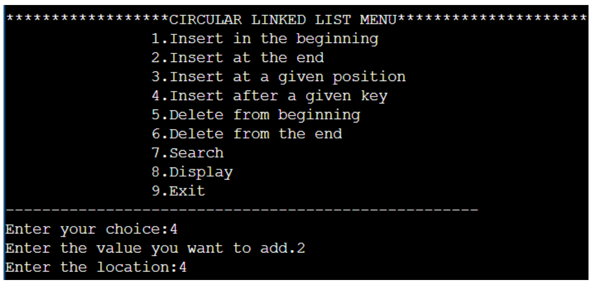
OUTPUT



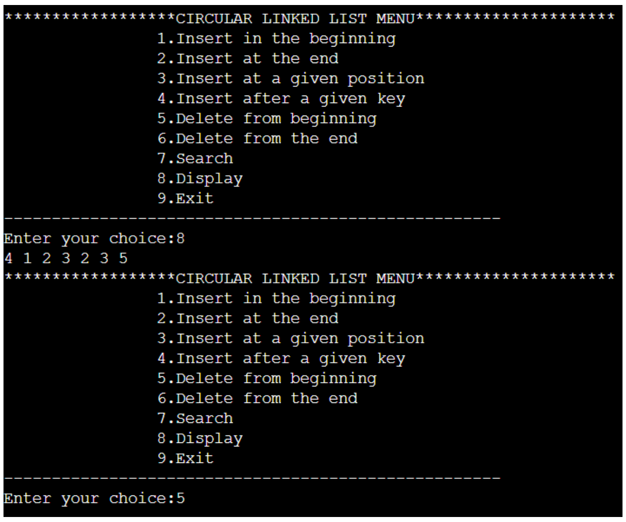
```
Enter number of elements: 3

Enter element to be inserted: 1

Enter element to be inserted: 2

Enter element to be inserted: 3

*****************CIRCULAR LINKED LIST MENU***********
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
-----------------------------------------------------
Enter your choice:1

Enter element to be inserted: 4
```



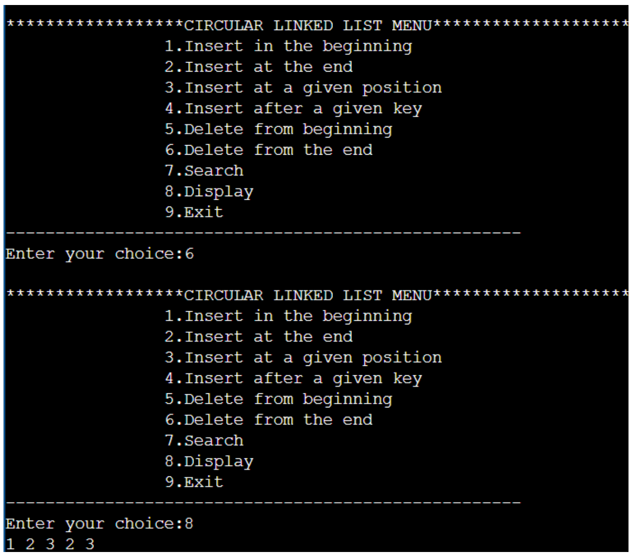
```
*****************CIRCULAR LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
-----------------------------------------------------
Enter your choice:2

Enter element to be inserted: 5

*****************CIRCULAR LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
-----------------------------------------------------
Enter your choice:3

Enter element to be inserted at the given position: 4

Enter the position: 3
```



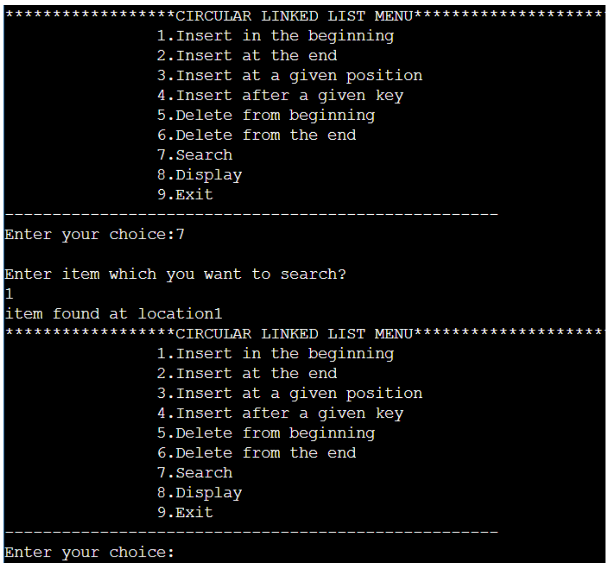
```
*****************CIRCULAR LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
-----------------------------------------------------
Enter your choice:4
Enter the value you want to add.2
Enter the location:4
```



```
*****************CIRCULAR LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
-----------------------------------------------------
Enter your choice:8
4 1 2 3 2 3 5
*****************CIRCULAR LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
-----------------------------------------------------
Enter your choice:5
```



```
*****************CIRCULAR LINKED LIST MENU*******************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:6

*****************CIRCULAR LINKED LIST MENU*******************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:8
1 2 3 2 3
```



```
*****************CIRCULAR LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:7

Enter item which you want to search?
1
item found at location1
*****************CIRCULAR LINKED LIST MENU*******************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the end
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:
```

Linear Linked List

Write a single menu driven program containing separate function for each of the following operation on singly linear Linked List. Ask user to execute functions of choice given in the menu. The function names must be same as below:

- **Insert_begin():** insert a new node at the beginning of the list.
- **Insert_last():** insert a new node at the last in the list.
- **Insert_random(pos):** insert a new node at the given position in the list.
- **Insert_specific(key):** insert a new node after the node containing given key.
- **Delete_begin():** delete node from beginning of the list.
- **Delete_end():** delete the last node from the list.
- **Search(key):** search the given key in the list and return the node's position containing the key.
- **Display():** print all the elements of the list.

```
#include <iostream>
using namespace std;


class Node
{
public:
  int data;
  Node *next;
};

void keyinsert(Node *head,Node *temp, int item)
{
    int i,loc;
    Node *tmp = head;
    if(tmp == NULL)
    {
        cout<<"\nOVERFLOW";
    }
    else
    {
        cout<<"Enter the location";
        cin>>loc;
        tmp->data = item;
        temp=head;
        for(i=0;i<loc;i++)
        {
            temp = temp->next;
            if(temp == NULL)
            {
                cout<<"\ncan't insert\n";
                return;
            }
        }
        tmp ->next = temp ->next;
        temp ->next = tmp;
        cout<<"\nNode inserted";
    }
}
```

```
void findNode(Node *head, int item)
{
        int index = 0;
        Node *tmp = head;
        while(tmp!=NULL)
        {
                if(tmp->data == item)
                {
                        cout<<index;
                }
                tmp = tmp->next;
                index++;
                cout<<"It is present at "<<index<< endl;
        }
        cout<<"Not present";
}

void display(Node *head)
{
  Node *tmp = head;
  while (tmp->next != NULL)
  {
    cout << tmp->data << " ";
    tmp = tmp->next;
  }
  cout << tmp->data << " ";
}

void insert_Atlast(Node **head_ref, int item)
```

```
{

  if (*head_ref == NULL)
  {
    //empty LL
    Node *new_node = NULL;
    new_node = new Node();
    new_node->data = item;
    new_node->next = NULL;
    *head_ref = new_node;
    return;
  }
  else
  {
    // non-empty LL
    Node *tmp = *head_ref;
    while (tmp->next != NULL)
    {
      tmp = tmp->next;
    }
    Node *new_node = new Node();
    new_node->data = item;
    new_node->next = NULL;
    tmp->next = new_node;
    return;
  }
}

void insert_Atfirst(Node **head_ref, int item)
{
  Node *new_node = new Node();
  new_node->data = item;
  new_node->next = *head_ref;
  *head_ref = new_node;
  return;
}

void insert_random(Node *head, int pos, int item)
{
  Node *tmp = head;
  int count = 1;
  while (count < pos - 1)
  {
    count++;
    tmp = tmp->next;
  }
  Node *new_node = new Node();
  new_node->data = item;
  new_node->next = tmp->next;
  tmp->next = new_node;
  return;
}

void delete_Atfirst(Node **head_ref)
{
  Node *tmp = *head_ref;
  *head_ref = tmp->next;
  tmp->next = NULL;
  return;
```

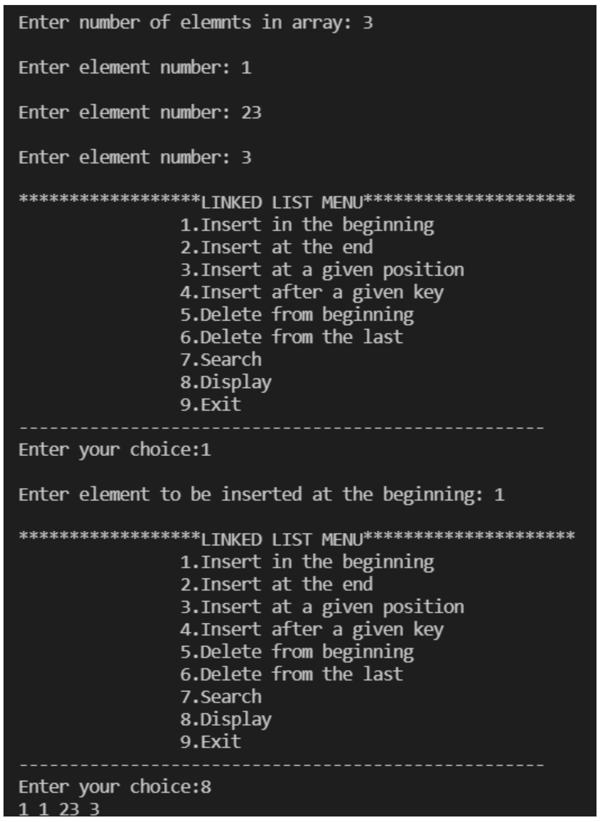
```
}

void delete_Atlast(Node **head_ref)
{
  if (*head_ref == NULL)
  {
    cout << "\nList is empty";
    return;
  }
  Node *tmp = *head_ref;
  Node *tail = NULL;
  while (tmp->next != NULL)
  {
    tail = tmp;
    tmp = tmp->next;
  }
  // cout<<tmp->data<<" ";
  tail->next = NULL;
  return;
}

int main()
{
  int ch = 0;
  int size;
  cout << "\nEnter number of elemnts in array: ";
  cin >> size;

  Node *head = NULL;
  Node *temp = NULL;
```

```
for (int i = 0; i < size; i++)
{
  int x, item;
  cout << "\nEnter element number: ";
  cin >> x;
  insert_Atlast(&head, x);
}
     while(ch!=9)
     {         cout<<"\n******************LINKED LIST MENU*********************\n";
               cout<<"\t\t1.Insert in the beginning\n";
               cout<<"\t\t2.Insert at the end\n";
               cout<<"\t\t3.Insert at a given position\n";
               cout<<"\t\t4.Insert after a given key\n";
               cout<<"\t\t5.Delete from beginning\n";
               cout<<"\t\t6.Delete from a specified position\n";
               cout<<"\t\t7.Search\n";
               cout<<"\t\t8.Display\n";
               cout<<"\t\t9.Exit\n";
               cout<<"---------------------------------------------------\n";
               cout<<"Enter your choice:";
               cin>>ch;
               if(ch==1)
               {   int item;
                   cout << "\nEnter element to be inserted at the beginning: ";
          cin >> item;
          insert_Atfirst(&head, item);
               }
               else if(ch==2)
               {   int item;
```

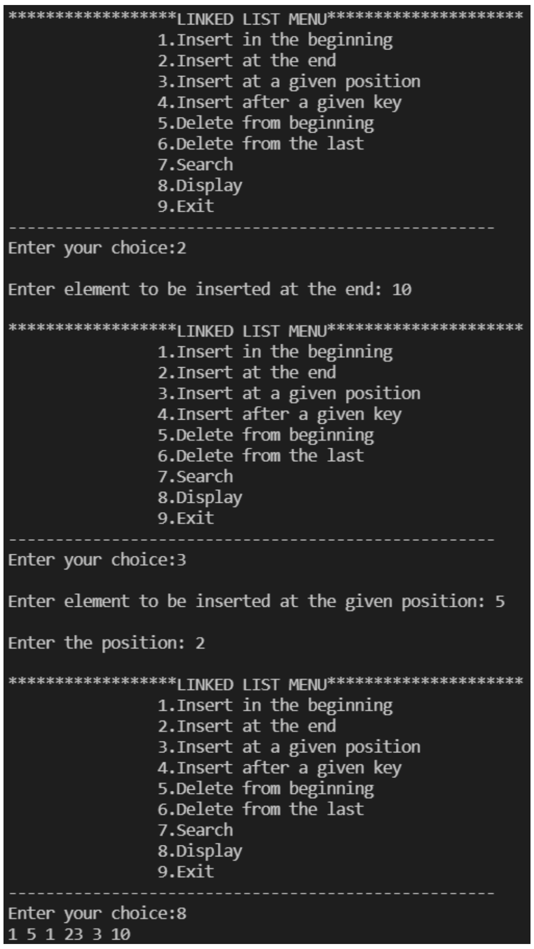
```
            cout << "\nEnter element to be inserted at the end: ";
cin >> item;
insert_Atlast(&head, item);
        }
        else if(ch==3)
        {   int item;
            cout << "\nEnter element to be inserted at the given position: ";
cin >> item;
int pos;
cout << "\nEnter the position: ";
insert_random(head, pos, item);
        }
        else if(ch==4)
        {   int item;

            keyinsert(head,temp,item);
        }
        else if(ch==5)
        {
            delete_Atfirst(&head);
        }
        else if(ch==6)
        {
            delete_Atlast(&head);
        }
        else if(ch==7)
        {
            int item;
    cout<<"Enter the data of the linked list to be found."<<endl;
    cin>>item;
```

```
            findNode(head,item);
              }
              else if(ch==8)
              {
                 display(head);
              }
        }


}
```

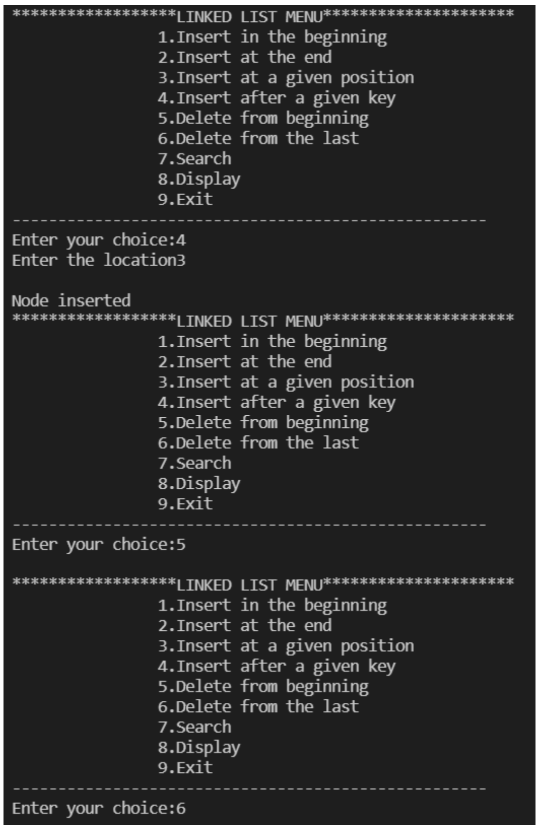
OUTPUT



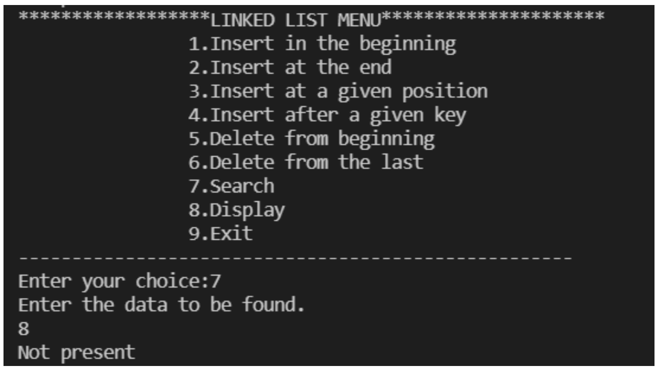
```
Enter number of elemnts in array: 3

Enter element number: 1

Enter element number: 23

Enter element number: 3

*****************LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:1

Enter element to be inserted at the beginning: 1

*****************LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:8
1 1 23 3
```



```
*****************LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:2

Enter element to be inserted at the end: 10

*****************LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:3

Enter element to be inserted at the given position: 5

Enter the position: 2

*****************LINKED LIST MENU********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
---------------------------------------------------
Enter your choice:8
1 5 1 23 3 10
```



```
******************LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:4
Enter the location3

Node inserted
******************LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:5

******************LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:6
```



```
******************LINKED LIST MENU*********************
                1.Insert in the beginning
                2.Insert at the end
                3.Insert at a given position
                4.Insert after a given key
                5.Delete from beginning
                6.Delete from the last
                7.Search
                8.Display
                9.Exit
----------------------------------------------------
Enter your choice:7
Enter the data to be found.
8
Not present
```

Array

Write a single menu driven program containing separate function for each of the following operations for linear array. Ask user to execute functions of choices given in the menu. The function names must be same as below:

- **Insert(item, index):** Insert the given item at the specified index.
- **Delete(index):** delete the item from the specified index.
- **Linear_search(key):** Search the given key in the array using linear search.
- **Binary_search(key):** Search the given key element in the array using binary search.
- **Display():** Print the array elements.

```
#include<iostream>

#include<stdio.h>

using namespace std;

int i, x, pos, size, a[50],num;
```

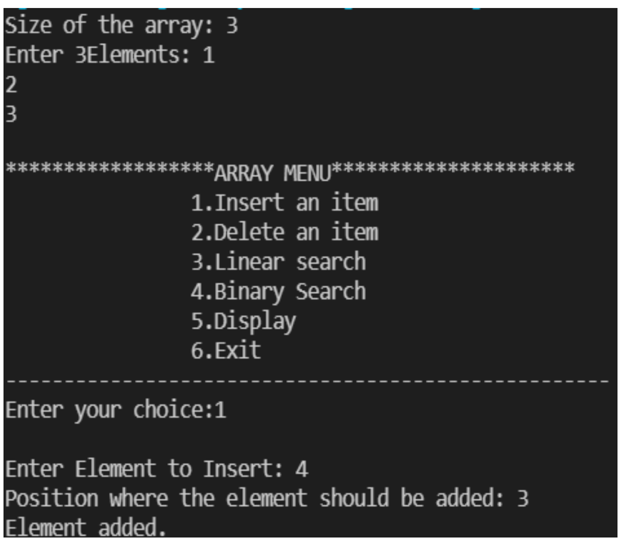
```
void insert()
{
   cout<<"\nEnter Element to Insert: ";
   cin>>x;
   cout<<"Position where the element should be added: ";
   cin>>pos;
   for(int i=size; i>=pos; i--)
   {
      a[i] = a[i-1];
   }
   a[pos-1] = x;
   size++;
   cout<<"Element added.";
}

void lsearch()
{   int index;
   cout<<"\nEnter a Number to Search: ";
   cin>>num;
   for(i=0; i<10; i++)
   {
      if(a[i]==num)
      {
         index = i;
         break;
      }
   }
   cout<<"\nFound at Index No."<<index;
   cout<<endl;
}
```

```
void del()
{
    cout << "\n\n Enter the position of the element to be deleted : ";
    cin >> pos;
    --pos;
    for (i = pos; i <= 9; i++)
    {
        a[i] = a[i + 1];
    }
}
void display()
{
    for(i=0; i<size; i++)
        cout<<a[i]<<" ";
}

int binsearch(int a[],int size,int item)
{   /* For sorting the array*/
    int temp;
    for(i=0;i<size;i++)
    {
        for(int j=0;j<size-1;j++)
        {
            if(a[j]>a[j+1])
                {
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                }
            }
        }
    }
    int mid;
    int beg=0;
    int last=size-1;
    while (beg<=last)

    {   mid=(beg+last)/2;
        if(item==arr[mid])
        {
            return mid;
        }
        else if(item>arr[mid])
        {
            beg=mid+1;
        }
        else
        {
            last=mid-1;
        }
    }
    return -1;
}




int main()
{   int num;
```

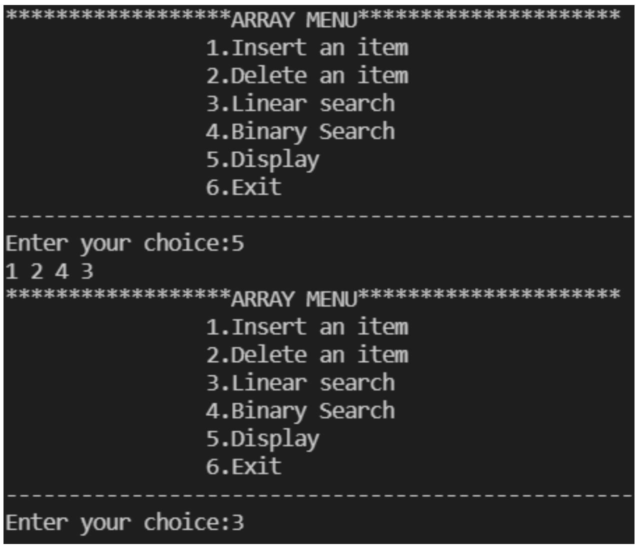
```
cout<<"Size of the array: ";
cin>>size;
cout<<"Enter "<<size<<"Elements: ";
for(i=0; i<size; i++)
{
    cin>>a[i];
}
   int ch=0;
   int value;
   while(ch!=7)
   {          cout<<"\n*****************ARRAY MENU********************\n";
              cout<<"\t\t1.Insert an item\n";
              cout<<"\t\t2.Delete an item\n";
              cout<<"\t\t3.Linear search\n";
              cout<<"\t\t4.Binary Search\n";
              cout<<"\t\t5.Display\n";
              cout<<"\t\t6.Exit\n";
              cout<<"--------------------------------------------------\n";
              cout<<"Enter your choice:";
              cin>>ch;
              if(ch==1)
              {
                  insert();
              }
              else if(ch==2)
              {
                  del();                  }
              else if(ch==3)
              {
                  lsearch();
```

```
            }
            else if(ch==4)
            {
              cout <<"\n\n\t\t\t Enter item to be searched  :";
        cin >>item;
        int flag=binsearch(a[50],size,item);
        if(flag==-1)
        {
          cout<<"\n\n\t\t\t The searched number is not available in the above array";
        }
        else
        {
          cout<<"\n\n\t\t\t The searched numbers is found at index value:"<<flag;
          cout<<"\n\n\t\t\t Position in the array is:"<<flag+1;
        }
            }
            else if(ch==5)
            {
              display();
            }
      }

}
```

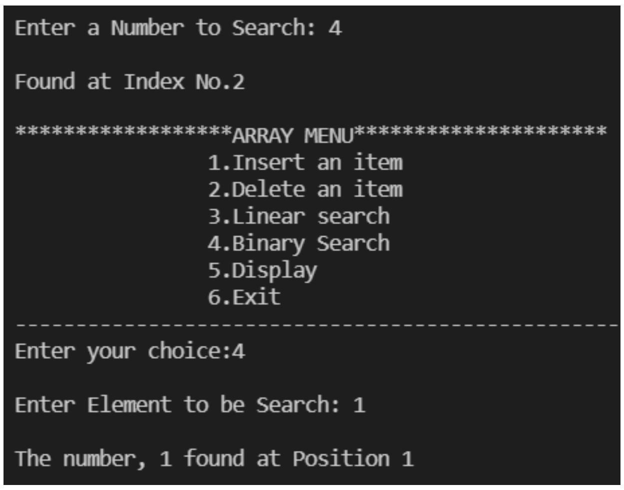
OUTPUT



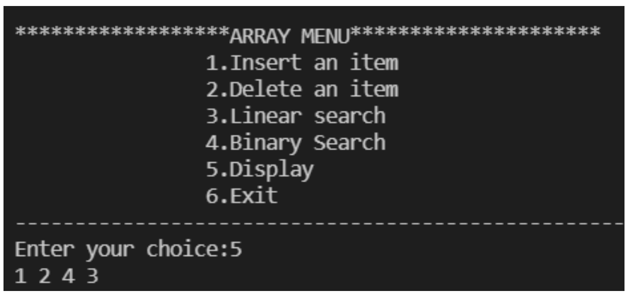
```
Size of the array: 3
Enter 3Elements: 1
2
3

*****************ARRAY MENU********************
                1.Insert an item
                2.Delete an item
                3.Linear search
                4.Binary Search
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:1

Enter Element to Insert: 4
Position where the element should be added: 3
Element added.
```



```
*****************ARRAY MENU********************
                1.Insert an item
                2.Delete an item
                3.Linear search
                4.Binary Search
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:5
1 2 4 3
*****************ARRAY MENU********************
                1.Insert an item
                2.Delete an item
                3.Linear search
                4.Binary Search
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:3
```



```
Enter a Number to Search: 4

Found at Index No.2

*****************ARRAY MENU********************
                1.Insert an item
                2.Delete an item
                3.Linear search
                4.Binary Search
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:4

Enter Element to be Search: 1

The number, 1 found at Position 1
```



```
*****************ARRAY MENU********************
                1.Insert an item
                2.Delete an item
                3.Linear search
                4.Binary Search
                5.Display
                6.Exit
---------------------------------------------------
Enter your choice:5
1 2 4 3
```