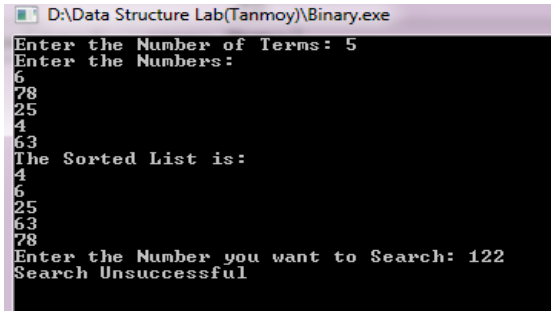
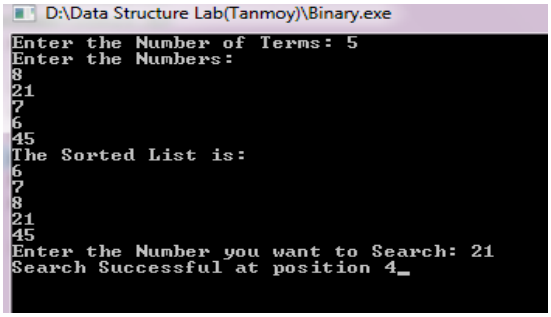
PROGRAM TO SEARCH AN ITEM FROM AN ARRAY USING BINARY SEARCH:

CODE:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[100],n,start,end,mid,i,s,j,swap;
   printf("Enter the Number of Terms: ");
   scanf("%d",&n);
   printf("Enter the Numbers: \n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
    for(i=0;i<n-1;i++)
   {
              for(j=0;j<n-1-i;j++)
              {
                        if(a[j]>a[j+1])
                        {
                                  swap=a[j];
                                  a[j]=a[j+1];
                                  a[j+1]=swap;
                        }
              }
   }
   printf("The Sorted List is: \n");
   for(i=0;i<n;i++)
   printf("%d\n",a[i]);
   printf("Enter the Number you want to Search: ");
   scanf("%d",&s);
   start=0;
   end=n-1;
   mid=(start+end)/2;
   while (start<=end)
   {
         if(a[mid]<s)
         start=mid+1;
         else if(a[mid]==s)
         {
         printf("Search Successful at position %d", mid+1);
         break;
         }
         else
         end=mid-1;
         mid=(start+end)/2;
   }
   if(start>end)
   printf("Search Unsuccessful");
   getch();
  }
```

OUTPUT:-



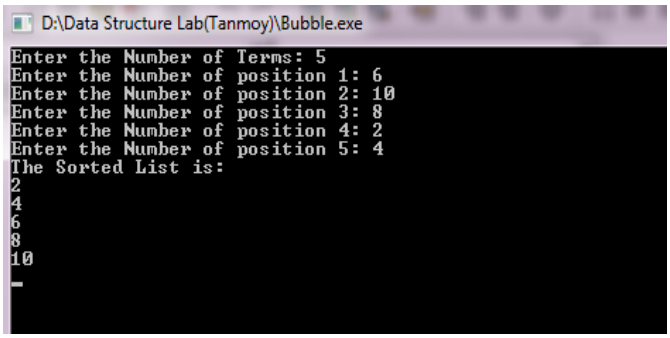


PROGRAM TO SORT AN ARRAY (BUBBLE SORT):

CODE:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[100],n,i,j,swap;
   printf("Enter the Number of Terms: ");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
            printf("Enter the Number of position %d: ",i+1);
            scanf("%d",&a[i]);
   }
   for(i=0;i<n-1;i++)
   {
             for(j=0;j<n-1-i;j++)
             {
                      if(a[j]>a[j+1])
                      {
                               swap=a[j];
                               a[j]=a[j+1];
                               a[j+1]=swap;
                      }
             }
   }
   printf("The Sorted List is: \n");
   for(i=0;i<n;i++)
   printf("%d\n",a[i]);
   getch();
   return 0;
}
```

OUTPUT:-

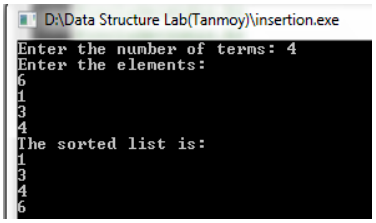


PROGRAM TO SORT AN ARRAY (INSERTION SORT):

CODE:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[100],n,i,j,key;
   printf("Enter the number of terms: ");
   scanf("%d",&n);
   printf("Enter the elements: \n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   for(j=1;j<=n-1;j++)
   {
                key=a[j];
                i=j-1;
                while(i>=0 && a[i]>key)
                {
                        a[i+1]=a[i];
                        i=i-1;
                }
                a[i+1]=key;
   }
   printf("The sorted list is:\n");
   for(i=0;i<n;i++)
   printf("%d\n",a[i]);
   getch();
   return 0;
}
```

OUTPUT:-

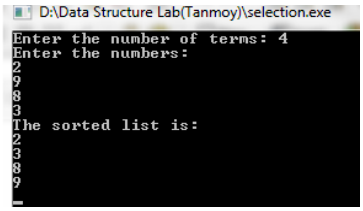


PROGRAM TO SORT AN ARRAY (SELECTION SORT):

CODE:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[100],n,i,j,small,swap;
   printf("Enter the number of terms: ");
   scanf("%d",&n);
   printf("Enter the numbers: \n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   for(j=0;j<n-1;j++)
   {
                small=j;
                for(i=j+1;i<n;i++)
                {
                        if(a[i]<a[small])
                        small=i;
                }
                swap=a[j];
                a[j]=a[small];
                a[small]=swap;
   }
   printf("The sorted list is:\n");
   for(i=0;i<n;i++)
   printf("%d\n",a[i]);
   getch();
   return 0;
}
```

OUTPUT:-

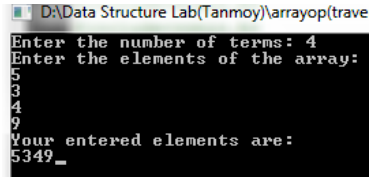


ARRAY OPERATION (TRAVERSING AN ARRAY):

CODE:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[20],i,n;
   printf("Enter the number of terms: ");
   scanf("%d",&n);
   printf("Enter the elements of the array:\n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   printf("Your entered elements are:\n");
   for(i=0;i<n;i++)
   printf("%d",a[i]);
   getch();
}
```

OUTPUT:-

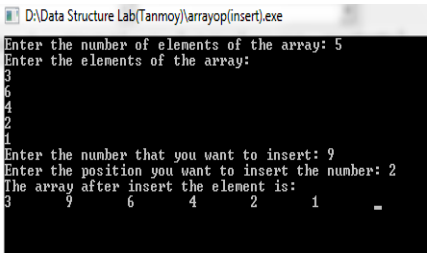


ARRAY OPERATION (INSERT AN ELEMENT IN THE ARRAY):

CODE:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[10],k,item,n,i=0,j;
   printf("Enter the number of elements of the array: ");
   scanf("%d",&n);
   j=n;
   printf("Enter the elements of the array: \n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   printf("Enter the number that you want to insert: ");
   scanf("%d",&item);
   printf("Enter the position you want to insert the number: ");
   scanf("%d",&k);
   k--;
   n=n+1;
    while(j>=k)
                {
                        a[j+1]=a[j];
                        j=j-1;
                }
                a[k]=item;
                printf("The array after insert the element is:\n");
               for(i=0;i<n;i++)
               printf("%d\t",a[i]);
   getch();
   return 0;
}
```

OUTPUT:-

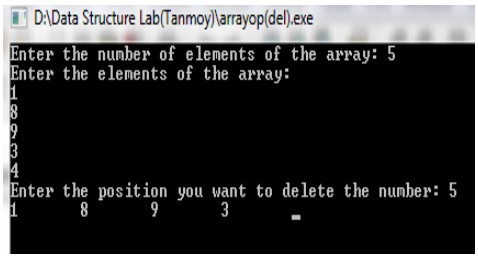


ARRAY OPERATION (DELETE AN ELEMENT FROM THE ARRAY):

CODE:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[10],k,n,i,j,item;
   printf("Enter the number of elements of the array: ");
   scanf("%d",&n);
   printf("Enter the elements of the array: \n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   printf("Enter the position you want to delete the number: ");
   scanf("%d",&k);
   item=a[k];
   j=k;
   n=n-1;
   while(j<k)
   {
         a[j]=a[j+1];
         j=j+1;
   }
   for(i=0;i<n;i++)
   printf("%d\t",a[i]);
   getch();
   return 0;
}
```

OUTPUT:-



Program for String operations without library functions:

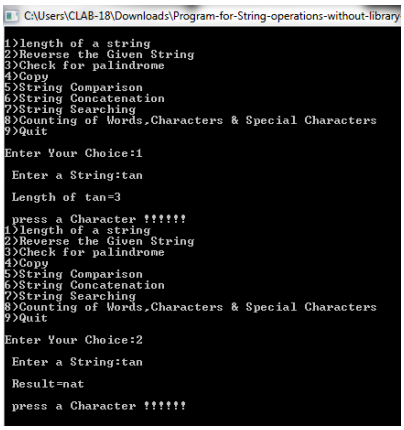
CODE:-

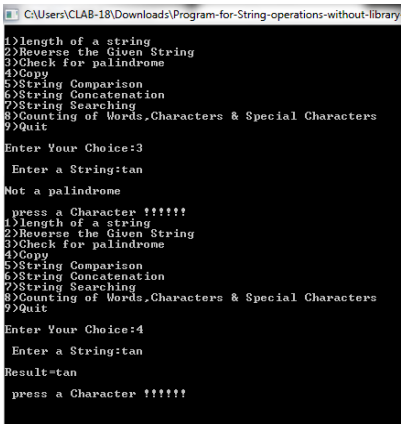
```
#include <stdio.h>
#include<string.h>
#include<conio.h>
#include<ctype.h>
#include<stdlib.h>
/*function prototypes*/
int length(char a[]);
void reverse(char a[]);
int palindrome(char a[]);
void copy(char a[],char b[]);
int compare(char a[],char b[]);
void concat(char a[],char b[]);
void search(char a[],char b[]);
void count(char a[]);

int main()
 { char a[100],b[100];
   int result,op;
   void clrscr();
   do
    {
          printf("\n1)length of a string");
          printf("\n2)Reverse the Given String");
          printf("\n3)Check for palindrome");
          printf("\n4)Copy");
          printf("\n5)String Comparison");
          printf("\n6)String Concatenation");
          printf("\n7)String Searching");
          printf("\n8)Counting of Words,Characters & Special Characters");
          printf("\n9)Quit");
          printf("\n\nEnter Your Choice:");
          scanf("%d",&op);
          switch(op)
           { case 1: printf("\n Enter a String:");
                     scanf("%s",&a);
                     result=length(a);
                     printf("\n Length of %s=%d",a,result);
                     printf("\n\n press a Character !!!!!!");
                     getch();
                     break;

             case 2: printf("\n Enter a String:");
                     scanf("%s",&a);
                     reverse(a);
                     printf("\n Result=%s",a);
```

```
            printf("\n\n press a Character !!!!!!");
            getch();
            break;
 case 3: printf("\n Enter a String:");
            scanf("%s",&a);
            result=palindrome(a);
            if(result==0)
              printf("\nNot a palindrome");
            else
              printf("\nA palindrome");
             printf("\n\n press a Character !!!!!!");
            getch();
            break;
case 4: printf("\n Enter a String:");
            scanf("%s",&a);
            copy(b,a);
            printf("\nResult=%s",b);
            printf("\n\n press a Character !!!!!!");
            getch();
            break;
case 5: printf("\n Enter 1st string:");
            scanf("%s",&a);
            printf("\n Enter 2nd string:");
            gets(b);
            result=compare(a,b);
            if(result==0)
              printf("\nboth are same");
            else
               if(result>0)
                     printf("\n1st>2nd");
                       else
                         printf("\n1st<2nd");
            printf("\n\n press a Character !!!!!!");
            getch();break;
case 6: printf("\n Enter 1st string:");
            scanf("%s",&a);
            printf("\n Enter 2nd string:");
            gets(b);
            concat(a,b);
            printf("\nresult=%s",a);
            printf("\n\n press a Character !!!!!!");
            getch();break;
case 7: printf("\n Enter 1st string:");
            scanf("%s",&a);
            printf("\n Enter 2nd string:");
            scanf("%s",&b);
            search(a,b);
            printf("\n\n press a Character !!!!!!");
```

```
                    getch();break;
            case 8: printf("\n Enter a string:");
                    scanf("%s",&a);
                    count(a);
                    printf("\n\n press a Character !!!!!!");
                    getch();break;
            default : printf("\n A wrong Choice:");break;
           }
    }while(op!=9);
 }

int length(char a[])
 { int i=0;
   while(a[i] !='\0')
    i++;
  return(i);
 }
void reverse(char a[])
  { int i,j;
    char temp;
     i=j=0;
     while(a[j]!='\0')
      j++;
     j--;
     while(i<j)
      { temp=a[i]; a[i]=a[j];a[j]=temp;
            i++;j--;
      }
    }
int palindrome(char a[])
 { int i,j;
    i=j=0;
     while(a[j]!='\0')
      j++;
     j--;
     while(i<j)
      { if(a[i]!=a[j])
             return(0);
            i++;j--;
      }
     return(1);
    }
 void copy(char b[],char a[])
    { int i=0;
      while(a[i]!='\0')
           { b[i]=a[i];
             i++;
           }
```

```
    b[i]='\0';
   }

 int compare(char a[],char b[])
   { int i;
   i=0;
   while(a[i]!='\0')
    { if(a[i] > b[i])
           return(1);
          if(a[i] < b[i])
           return(-1);
          i++;
    }
    return(0);
  }
 void concat(char a[],char b[])
  { int i,j;
   i=0;
   while(a[i]!='\0')
    i++;
   for(j=0;b[j]!='\0';i++,j++)
     a[i]=b[j];
   a[i]='\0';
 }
void search(char a[] ,char b[])
 { int i,j,lena,lenb;
  for(lena=0;a[lena]!='\0';lena++);
  for(lenb=0;b[lenb]!='\0';lenb++);
  for(i=0;i<=lena-lenb+1;i++)
   {
   for(j=0;a[i+j]==b[j]&&b[j]!='\0';j++);
          if(b[j]=='\0')
            printf("\nstring found at location:%d",i+1);
   }
 }
 void count(char a[])
 { int words=0,characters=0,spchar=0,i;
   for(i=0;a[i]!='\0';i++)
    { if(isalnum(a[i]) && (i==0 || !isalnum(a[i-1])))
           words++;
     characters++;
     if(!isalnum(a[i]) && !isspace(a[i]))
           spchar++;
   }
   printf("\n no of characters=%d",characters);
   printf("\n no of special characters=%d",spchar);
   printf("\n no of words=%d",words);
 }
```





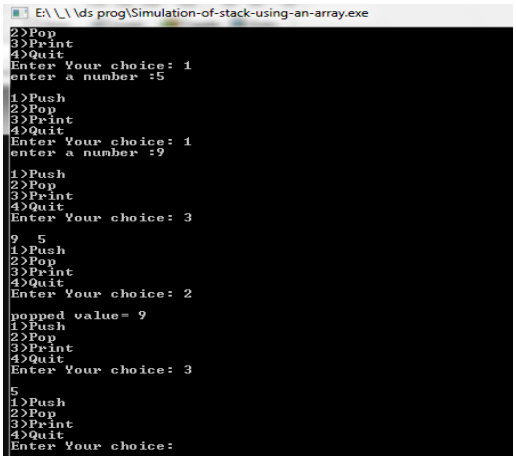
SIMULATION OF STACK USING AN ARRAY:

CODE:-

```
#include<stdio.h>
#include<conio.h>
#define MAX 6
typedef struct stack
{
        int data[MAX];
        int top;
}stack;
void init(stack *);
int empty(stack *);
int full(stack *);
int pop(stack *);
void push(stack *,int);
void print(stack *);
void main()
{
        stack s;
        int x,op;
        init(&s);
        clrscr();
        do {
            printf("\n\n1)Push\n2)Pop\n3)Print\n4)Quit");
            printf("\nEnter Your choice: ");
            scanf("%d",&op);
            switch(op)
             {
                case 1:printf("\n enter a number :");
                    scanf("%d",&x);
                    if(!full(&s))
                        push(&s,x);
                    else
                        printf("\nStack is full.....");
                    break;
                case 2:if(!empty(&s))
                        { x=pop(&s);
                          printf("\npopped value= %d",x);
                        }
                    else
                       printf("\nStack is empty.....");
                    break;

                case 3:print(&s);break;
                default: prinyf("wrong input");
            }
    }while(op!=4);   //while(op<=0 || op>=5);
```

```
}
void init(stack *s)
{
        s->top=-1;
}

int empty(stack *s)
{
        if(s->top==-1)
                return(1);
        return(0);
}

int full(stack *s)
{
        if(s->top==MAX-1)
                return(1);
        return(0);
}

void push(stack *s,int x)

{
        s->top=s->top+1;
        s->data[s->top]=x;
}

int pop(stack *s)
{
        int x;
        x=s->data[s->top];
        s->top=s->top-1;
        return(x);
}
void print(stack *s)
 {
        int i;
        printf("\n");
        for(i=s->top;i>=0;i--)
                printf("%d  ",s->data[i]);
 }
```

OUTPUT:-



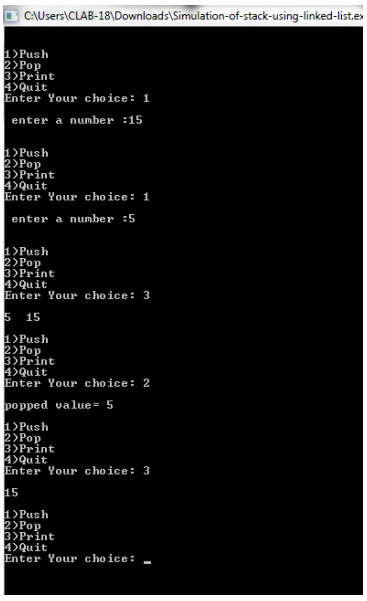
SIMULATION OF STACK USING A LINKED LIST:

CODE:-

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
        int data;
        struct node *next;
} node;
node * init();
int empty(node *top);
int gettop(node *top);
node *  Delete(node *);
node *  insert(node *top ,int x);
void print(node  *top);
int main()
{
        node *top;
        int x,op;
        top=init();
        void clrscr();
        do {
            printf("\n\n1)Push\n2)Pop\n3)Print\n4)Quit");
            printf("\nEnter Your choice: ");
            scanf("%d",&op);
            switch(op)
             { case 1:printf("\n enter a number :");
                        scanf("%d",&x);
                        top=insert(top,x);
                        break;
              case 2:if(!empty(top))
                                {
                                        x=gettop(top);
                                        printf("\npopped value= %d",x);
                                        top=Delete(top);
                                }
                        else
                                        printf("\nStack is empty.....");
                        break;
              case 3:print(top);break;
            }
    }while(op!=4);
}
node * init()
{
    return(NULL);
```

```
}
int empty(node  *top)
{
        if(top==NULL)
                return(1);
        return(0);
}
node * insert(node *top,int x)
{
        node  *p;
        p=(node*)malloc(sizeof(node));
        p->data=x;
        p->next=top;
        return(p);
}
int gettop(node *top)
 {
  int x;
  x=top->data;
  return(x);
 }
node* Delete(node *top)
{
        node *p;
        p=top;
        top=top->next;
        free(p);
        return(top);
}
void print(node *p)
 {
        printf("\n");
        while(p!=NULL)
          {
                printf("%d  ",p->data);
                p=p->next;
          }
}
```

OUTPUT:-



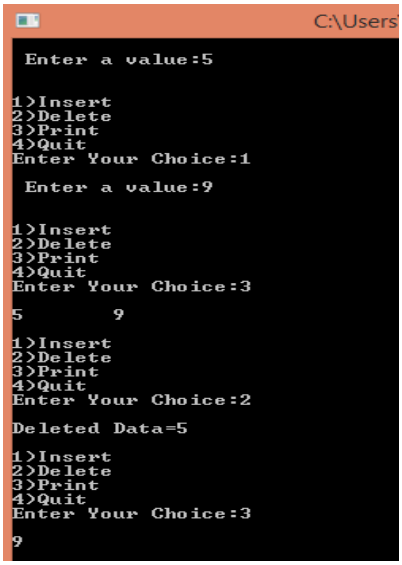
SIMULATION OF SINGLE QUEUE USING ARRAY:

CODE:-

```
#include<conio.h>
#include<stdio.h>
#define MAX 5
typedef struct Q
{
        int R,F;
        int data[MAX];
}Q;
void initialise(Q *P);
int empty(Q *P);
int full(Q *P);
void enqueue(Q *P,int x);
int dequeue(Q *P);
void print(Q *P);
int main()
{
        Q q;
        int op,x;
        initialise(&q);
        void clrscr();
        do
          {
                printf("\n\n1)Insert\n2)Delete\n3)Print\n4)Quit");
                printf("\nEnter Your Choice:");
                scanf("%d",&op);
                switch(op)
                  {
                        case 1: printf("\n Enter a value:");
                                scanf("%d",&x);
                                if(!full(&q))
                                        enqueue(&q,x);
                                else
                                        printf("\nQueue is full !!!!");
                                break;
                        case 2: if(!empty(&q))
                                  {
                                        x=dequeue(&q);
                                        printf("\nDeleted Data=%d",x);
                                  }
                                else
                                printf("\nQueue is empty !!!!");
                                break;
                        case 3: print(&q);break;
                  }
    }while(op!=4);
    return 0;
}
void initialise(Q *P)
{
        P->R=-1;
```

```
        P->F=-1;
}
int empty(Q *P)
{
        if(P->R==-1)
                return(1);
        return(0);
}
int full(Q *P)
{
        if(P->R==MAX-1)
                return(1);
        return(0);
}
void enqueue(Q *P,int x)
{
        if(P->R==-1)
        {
                P->R=P->F=0;
                P->data[P->R]=x;
        }
        else
        {
                P->R=P->R+1%MAX;
                P->data[P->R]=x;

        }
}
int dequeue(Q *P)
{
        int x;
        x=P->data[P->F];
        if(P->R==P->F)
        {
                P->R=-1;
                P->F=-1;
        }
        else
                P->F=P->F+1;
        return(x);
}
void print(Q *P)
{
        int i;
        if(!empty(P))
        {
                printf("\n");
                for(i=P->F;i<=P->R;i++)
                        printf("%d\t",P->data[i]);
        }
}
```

OUTPUT:-



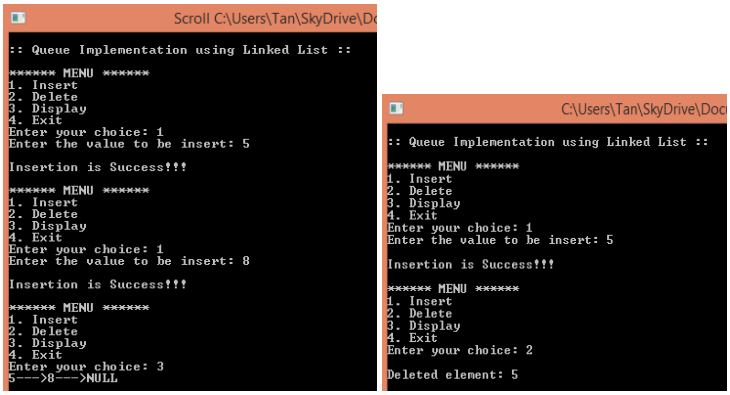
SIMULATION OF SINGLE QUEUE USING LINKED LIST:

CODE:-

```
#include<stdio.h>
#include<conio.h>
struct Node
{
  int data;
  struct Node *next;
}*front = NULL,*rear = NULL;
void insert(int);
void delete();
void display();
int main()
{
  int choice, value;
  void clrscr();
  printf("\n:: Queue Implementation using Linked List ::\n");
  while(1){
    printf("\n****** MENU ******\n");
    printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);
    switch(choice){
          case 1: printf("Enter the value to be insert: ");
                  scanf("%d", &value);
                  insert(value);
                  break;
          case 2: delete(); break;
          case 3: display(); break;
          case 4: exit(0);
          default: printf("\nWrong selection!!! Please try again!!!\n");
    }
  }
  return 0;
}
void insert(int value)
{
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = value;
  newNode -> next = NULL;
  if(front == NULL)
    front = rear = newNode;
  else{
    rear -> next = newNode;
    rear = newNode;
  }
```

```
  printf("\nInsertion is Success!!!\n");
}
void delete()
{
  if(front == NULL)
    printf("\nQueue is Empty!!!\n");
  else{
    struct Node *temp = front;
    front = front -> next;
    printf("\nDeleted element: %d\n", temp->data);
    free(temp);
  }
}
void display()
{
  if(front == NULL)
    printf("\nQueue is Empty!!!\n");
  else{
    struct Node *temp = front;
    while(temp->next != NULL){
          printf("%d--->",temp->data);
          temp = temp -> next;
    }
    printf("%d--->NULL\n",temp->data);
  }
}
```

OUTPUT:-



SIMULATION OF DE-QUEUE USING ARRAY:

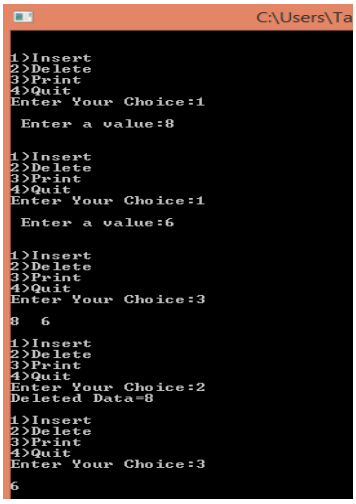
CODE:-

```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
typedef struct node
{
        int data;
        struct node *next;
}node;
node * initialise();
int empty(node *rear);
int full(node *rear);
node * insert(node *rear,int x);
node * Delete(node *rear);
int getfront(node *rear);
void print(node *rear);
int main()
{
        node *rear;
        int x,i,op;
        rear=initialise();
        void clrscr();
        do
          {
                printf("\n\n1)Insert\n2)Delete\n3)Print\n4)Quit");
                printf("\nEnter Your Choice:");
                scanf("%d",&op);
                switch(op)
                  {
                        case 1: printf("\n Enter a value:");
                                scanf("%d",&x);
                                rear=insert(rear,x);
                                break;
                        case 2: if(!empty(rear))
                                  {
                                        x=getfront(rear);
                                        printf("\Deleted Data=%d",x);
                                        rear=Delete(rear);
                                  }
                                else
                                        printf("\nQueue is empty !!!!");
                                break;
                        case 3: print(rear);break;
                  }
        }while(op!=4);
```

```
        return 0;
}
node * initialise()
{
   return(NULL);
}
node *insert(node *rear,int x)
{
        node *p;
        p=(node*)malloc(sizeof(node));
        p->data=x;
        if(rear==NULL)
        {
                p->next=p;
                return p;
        }
        else
        {
                p->next=rear->next;
                rear->next=p;
                return(p);
        }
}
node * Delete(node *rear)
{
        node *p;
        p=rear->next;
        if(rear->next==rear)
                rear=NULL;
        else
                rear->next=p->next;
        free(p);
        return(rear);
}
int getfront(node *rear)
 {
   return(rear->next->data);
 }
void print(node *rear)
{
        int i;
        node *p;
        if(rear!=NULL)
          {    printf("\n");
                p=rear->next;
                do
                 {
                        printf("%d  ",p->data);
```

```
                        p=p->next;
                    }while(p!=rear->next);
            }
}
int empty(node *rear)
{
        if(rear==NULL)
                return 1;
        return 0;
}
```

OUTPUT:-



SIMULATION OF CIRCULER QUEUE USING ARRAY:

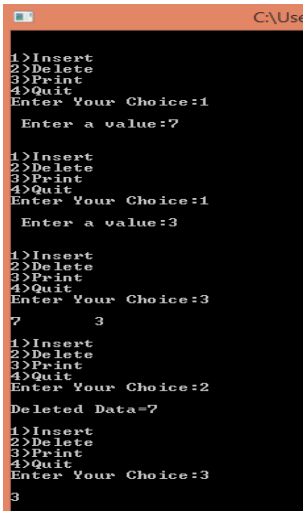
CODE:-

```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
typedef struct Q
{
        int R,F;
        int data[MAX];
}Q;
void initialise(Q *P);
int empty(Q *P);
int full(Q *P);
void enqueue(Q *P,int x);
int dequeue(Q *P);
void print(Q *P);
void main()
{
        Q q;
        int op,x;
        initialise(&q);
        void clrscr();
        do
          {
                printf("\n\n1)Insert\n2)Delete\n3)Print\n4)Quit");
                printf("\nEnter Your Choice:");
                scanf("%d",&op);
                switch(op)
                  {
                        case 1: printf("\n Enter a value:");
                                scanf("%d",&x);
                                if(!full(&q))
                                        enqueue(&q,x);
                                else
                                        printf("\nQueue is full !!!!");
                                break;
                        case 2: if(!empty(&q))
                                  {
                                        x=dequeue(&q);
                                        printf("\nDeleted Data=%d",x);
                                  }
                                else
                                printf("\nQueue is empty !!!!");
                                break;
                        case 3: print(&q);break;
                  }
```

```
    }while(op!=4);
}
void initialise(Q *P)
{
        P->R=-1;
        P->F=-1;
}
int empty(Q *P)
{
        if(P->R==-1)
                return(1);
        return(0);
}
int full(Q *P)
{
        if((P->R+1)%MAX==P->F)
                return(1);
        return(0);
}
void enqueue(Q *P,int x)
{
        if(P->R==-1)
        {
                P->R=P->F=0;
                P->data[P->R]=x;
        }
        else
        {
                P->R=(P->R+1)%MAX;
                P->data[P->R]=x;
        }
}
int dequeue(Q *P)
{
        int x;
        x=P->data[P->F];
        if(P->R==P->F)
        {
                P->R=-1;
                P->F=-1;
        }
        else
                P->F=(P->F+1)%MAX;
        return(x);
}
void print(Q *P)
{
        int i;
```

```
        if(!empty(P))
        {
                printf("\n");
                for(i=P->F;i!=P->R;i=(i+1)%MAX)
                        printf("%d\t",P->data[i]);
                printf("%d\t",P->data[i]);
        }
}
```

OUTPUT:-



SIMULATION OF SINGLE LINKED LIST:

CODE:-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct node
  { int data;
    struct node *next;
  }node;
node *create();
node *insert_b(node *head,int x);
node *insert_e(node *head,int x);
node *insert_in(node *head,int x);
node *delete_b(node *head);
node *delete_e(node *head);
node *delete_in(node *head);
node *reverse(node *head);
void search(node *head);
void print(node *head);
int main()
{ int op,op1,x;
  node *head=NULL;
  void clrscr();
```

```
 do
  {
    printf("\n\n1)create\n2)Insert\n3)Delete\n4)Search");
    printf("\n5)Reverse\n6)Print\n7)Quit");
    printf("\nEnter your Choice:");
    scanf("%d",&op);
    switch(op)
     { case 1:head=create();break;
           case 2:printf("\n\t1)Beginning\n\t2)End\n\t3)Anywhere");
                  printf("\nEnter your choice : ");
                  scanf("%d",&op1);
                  printf("\nEnter the data to be inserted : ");
                  scanf("%d",&x);
                  switch(op1)
                   {  case 1: head=insert_b(head,x);
                              break;
                     case 2: head=insert_e(head,x);
                              break;
                     case 3: head=insert_in(head, x);
                              break;
                    }
                  break;
           case 3:printf("\n\t1)Beginning\n\t2)End\n\t3)Anywhere");
                  printf("\nEnter your choice : ");
                  scanf("%d",&op1);
                  switch(op1)
                   {  case 1:head=delete_b(head);
                              break;
                     case 2:head=delete_e(head);
                              break;
                     case 3:head=delete_in(head);
                              break;
                    }
                   break;
           case 4:search(head);break;
           case 5:head=reverse(head);
                  print(head);
                  break;
           case 6:print(head);break;
      }
   }while(op!=7);
   return 0;
}
node *create()
{ node *head,*p;
  int i,n;
  head=NULL;
  printf("\n Enter no of data:");
```

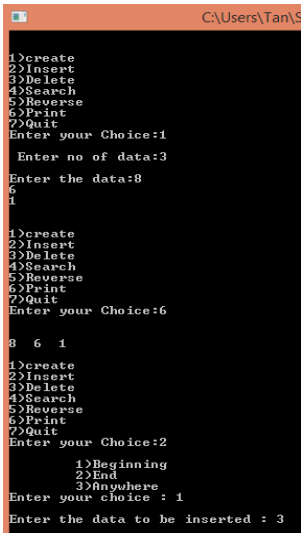
```
  scanf("%d",&n);
  printf("\nEnter the data:");
  for(i=0;i<n;i++)
   {
     if(head==NULL)
              p=head=(node*)malloc(sizeof(node));
     else
       {
              p->next=(node*)malloc(sizeof(node));
              p=p->next;
       }
       p->next=NULL;
       scanf("%d",&(p->data));
   }
 return(head);
}

node *insert_b(node *head,int x)
{   node *p;
    p=(node*)malloc(sizeof(node));
    p->data=x;
    p->next=head;
    head=p;
    return(head);
}
node *insert_e(node *head,int x)
{   node *p,*q;
    p=(node*)malloc(sizeof(node));
    p->data=x;
    p->next=NULL;
    if(head==NULL)
       return(p);
    //locate the last node
    for(q=head;q->next!=NULL;q=q->next)
    ;
    q->next=p;
    return(head);
}
node *insert_in(node *head,int x)
{           node *p,*q;
            int loc,i;
            p=(node*)malloc(sizeof(node));
            p->data=x;
            p->next=NULL;
            printf("\Enter the location : ");
            scanf("%d",&loc);
            if(loc==1) // inserting as a first node
              {
```

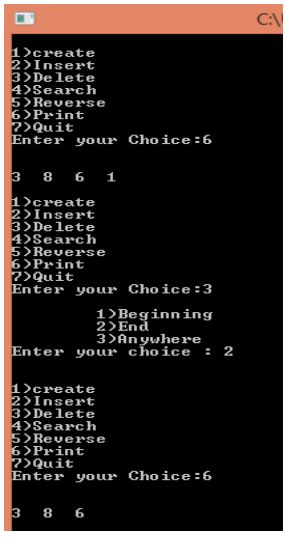
```
                p->next=head;
                head=p;
            }
          else
            {
                q=head;
                for(i=1; i<loc-1;i++)
                        if(q->next != NULL)
                                q=q->next;
                        else
                          {
                                printf("\nOverflow ****\n ");
                                return head;
                          }
                p->next=q->next;
                q->next=p;
            }
 return(head);
 }
node *delete_b(node *head)
{
  node *p,*q;
  if(head==NULL)
    {
          printf("\nUnderflow....Empty Linked List");
          return(head);
    }
  p=head;
  head=head->next;
  free(p);
  return(head);
}
node *delete_e(node *head)
{
  node *p,*q;
  if(head==NULL)
    {
          printf("\nUnderflow....Empty Linked List");
          return(head);
    }
  p=head;
  if(head->next==NULL)
    {
      head=NULL;
      free(p);
      return(head);
    }
   for(q=head;q->next->next !=NULL;q=q->next)
```

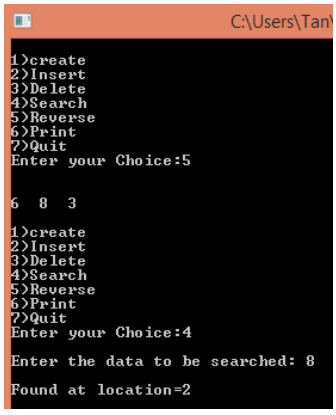
```
  ;
  p=q->next;
  q->next=NULL;
  free(p);
  return(head);
}
node *delete_in(node *head)
{
 node *p,*q;
 int loc,i;
 if(head==NULL)
   {
        printf("\nUnderflow....Empty Linked List");
        return(head);
   }
 printf("\nEnter the location of the data to be deleted : ");
 scanf("%d",&loc);
 if(loc==1)
   {
     p=head;
     head=head->next;
     free(p);
     return(head);
   }
  else
   {
        q=head;
        for(i=1; i<loc-1;i++)
                if(q->next->next!=NULL)
                                q=q->next;
                 else
                        {
                                printf("\nunderflow **** ");
                                return head;
                        }
        p=q->next;
        q->next=p->next;
        free(p);
   }
 return(head);

}

void search(node *head)
{ node *p;
 int data,loc=1;
 printf("\nEnter the data to be searched: ");
 scanf("%d",&data);
```

```
 p=head;
 while(p!=NULL && p->data != data)
  { loc++;
    p=p->next;
  }
 if(p==NULL)
  printf("\nNot found:");
 else
  printf("\nFound at location=%d",loc);
}
void print(node *head)
{ node *p;
 printf("\n\n");
 for(p=head;p!=NULL;p=p->next)
  printf("%d  ",p->data);
}
node *reverse(node *head)
 { node *p,*q,*r;
   p=NULL;
   q=head;
   r=q->next;
   while(q!=NULL)
    {
          q->next=p;
          p=q;
          q=r;
          if(q!=NULL)
            r=q->next;
    }
  return(p);
 }
```

OUTPUT:-







SIMULATION OF DOUBLY LINKED LIST:

CODE:-

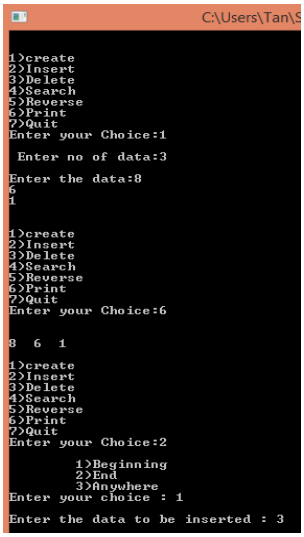
```
#include <stdio.h>
#include <conio.h>
#include<stdlib.h>
typedef struct  dnode
 {
     int  data;
     struct dnode *next,*prev;

  }dnode;
dnode * create()
 {
          int x,n,i;
          dnode *head=NULL;
          dnode *p;
          printf("\n Enter no of data : ");
          scanf("%d",&n);
          for(i=0;i<n;i++)
            {
                    printf("\n Next Data : ");
                    scanf("%d",&x);
                    if(head==NULL)
                       {
                         head=p=(dnode *) malloc(sizeof(dnode));
                         p->next=p->prev=NULL;
                         p->data=x;
                       }
                    else
                       {
                            p->next=(dnode *) malloc(sizeof(dnode));
                            p->next->data=x;
                            p->next->prev=p;
                            p=p->next;
                            p->next=NULL;
                       }
            }
  return(head);
  }
void print(dnode *head)
{
          dnode *p;
          printf("\n Data stored in the Doubly linked list : ");
          for(p=head; p!=NULL ; p=p->next)
                    printf("%d   ",p->data);
 }
 int search(dnode *head,int x)
```

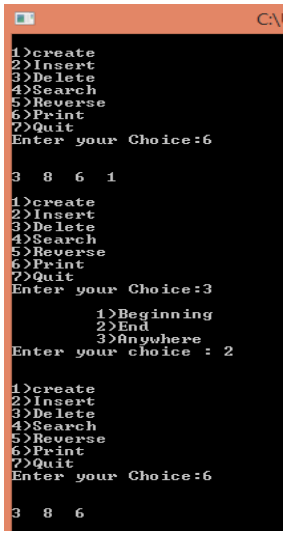
```
{
        int i=0;
        dnode *p;
        for(p=head; p!=NULL ; p=p->next)
          {
                if(p->data == x)
                        return(i);
                i++;
          }
        return -1;
}
dnode *insert(dnode *head,int x,int loc)
{
        dnode *p,*q;
        int i;
        p=(dnode*)malloc(sizeof(dnode));
        p->data=x;
        p->next=p->prev=NULL;
        if(loc==1)
          {
                p->next=head;
                head->prev=p;
                head=p;
          }
        else
          {
                q=head;
                for(i=1; i<loc-1;i++)
                        if(q->next!=NULL)
                                q=q->next;
                        else
                          {
                                printf("\nOverflow **** ");
                                return head;
                          }
                p->next=q->next;
                p->prev=q;
                q->next=p;
                q->next->prev=q;
          }
return(head);
}

dnode *Delete(dnode *head,int loc)
{
        dnode *p,*q;
        int i;
        if(loc==1)
```

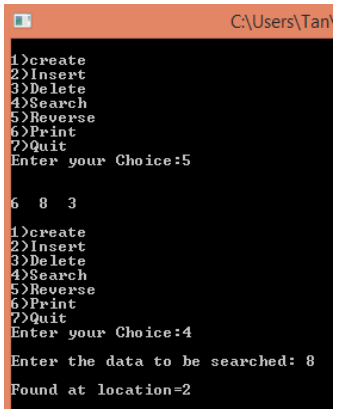
```
        {
                p=head;
                head=head->next;
                head->prev=NULL;
                free(p);
        }
       else
        {
                q=head;
                for(i=1;i<loc;i++)
                  {
                        if(q->next==NULL)
                          {
                                printf("\n Undeflow *****");
                                return(head);
                          }
                        else
                                q=q->next;
                   }
                if(q->next != NULL)
                        q->next->prev=q->prev;
                q->prev->next=q->next;
                free(q);
        }
 return(head);
 }
void main()
{
        int op,x,loc;
        dnode *head=NULL;
        void clrscr();
        do
         {
                printf("\n\n1)Create\n2)Print\n3)Insert\n4)Delete\n5)Search");
                printf("\n6)Quit");
                printf("\nEnter your Choice : ");
                scanf("%d",&op);
                switch(op)
                 {
                        case 1: head=create();break;
                        case 2: print(head);break;
                        case 3: printf("Enter the location :");
                                scanf("%d",&loc);
                                printf("Enter the data  :");
                                scanf("%d",&x);
                                head=insert(head,x,loc);break;
                        case 4: printf("Enter the location :");
                                scanf("%d",&loc);
```

```
                    head=Delete(head,loc);break;
                case 5: printf("Enter element to be searched : ");
                    scanf("%d",&x);
                    loc=search(head,x);
                    if(loc==-1)
                        printf("\n Element not found ");
                    else
                        printf("\n Found at location :%d ",loc+1);
                    break;
                }
        }while(op!=6);
  }
```

OUTPUT:-







SIMULATION OF CIRCULER LINKED LIST:

CODE:-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct node
  { int data;
    struct node *next;
  }node;

node *create();
node *insert_b(node *head,int x);
node *insert_e(node *head,int x);
node *insert_in(node *head,int x);
node *delete_b(node *head);
node *delete_e(node *head);
node *delete_in(node *head);
void search(node *head);
```

```
void print(node *head);
void main()
{ int op,op1,x;
  node *head=NULL;
  void clrscr();
  do
   {
    printf("\n\n1)create\n2)Insert\n3)Delete\n4)Search");
    printf("\n5)Print\n6)Quit");
    printf("\nEnter your Choice:");
    scanf("%d",&op);
    switch(op)
     { case 1:head=create();break;
            case 2:printf("\n\t1)Beginning\n\t2)End\n\t3)Anywhere");
                   printf("\nEnter your choice : ");
                   scanf("%d",&op1);
                   printf("\nEnter the data to be inserted : ");
                   scanf("%d",&x);
                   switch(op1)
                    {  case 1: head=insert_b(head,x);
                               break;
                      case 2: head=insert_e(head,x);
                               break;
                      case 3: head=insert_in(head, x);
                               break;
                     }
                   break;
            case 3:printf("\n\t1)Beginning\n\t2)End\n\t3)Anywhere");
                   printf("\nEnter your choice : ");
                   scanf("%d",&op1);
                   switch(op1)
                    {  case 1:head=delete_b(head);
                               break;
                      case 2:head=delete_e(head);
                               break;
                      case 3:head=delete_in(head);
                               break;
                     }
                    break;
            case 4:search(head);break;
            case 5:print(head);break;
      }
   }while(op!=6);
}
node *create()
{ node *head,*p;
  int i,n,x;
  head=NULL;
```

```
  printf("\n Enter no of data:");
  scanf("%d",&n);
  printf("\nEnter the data:");
  for(i=0;i<n;i++)
   {
     scanf("%d",&x);
     if(head==NULL)
       {
                        head=(node*)malloc(sizeof(node));
                        head->next=head;
                        head->data=x;
       }
     else
       {
                        p=(node*)malloc(sizeof(node));
                        p->data=x;
                        p->next=head->next;
                        head->next=p;
                        head=p;
       }
   }
 return(head);
}

node *insert_b(node *head,int x)
{   node *p;
    p=(node*)malloc(sizeof(node));
    p->data=x;
    if(head==NULL)
       {
                        p->next=p;
                        head=p;
       }
    else
       {
                        p->next=head->next;
                        head->next=p;
       }
    return(head);
}
node *insert_e(node *head,int x)
{   node *p;
    p=(node*)malloc(sizeof(node));
    p->data=x;
    if(head==NULL)
       {
                        p->next=p;
                        head=p;
```

```
    }
   else
    {
                p->next=head->next;
                head->next=p;
                head=p;
    }
   return(head);
}
node *insert_in(node *head,int x)
{       node *p,*q;
        int loc,i;
        p=(node*)malloc(sizeof(node));
        p->data=x;
        p->next=p;
        printf("\Enter the location : ");
        scanf("%d",&loc);
        if(loc==1)
          {   if(head==NULL)
                        return(p);
              else
               {
                        p->next=head->next;
                        head->next=p;
               }
          }
        else
          {
              q=head->next;
              for(i=1; i<loc-1;i++)
                        if(q->next != head->next)
                                q=q->next;
                        else
                          {
                                printf("\nOverflow ****\n ");
                                return head;
                          }
              p->next=q->next;
              q->next=p;
              if(q==head)
                head=p;
          }
 return(head);
 }
node *delete_b(node *head)
{
  node *p,*q;
  if(head==NULL)
```

```
    {
            printf("\nUnderflow....Empty Linked List");
            return(head);
    }
  p=head->next;
  if(head->next==head)
            head=NULL;
  else
            head->next=p->next;
  free(p);
  return(head);

}
node *delete_e(node *head)
{
  node *p,*q;
  if(head==NULL)
    {
            printf("\nUnderflow....Empty Linked List");
            return(head);
    }
  p=head->next;
  if(head->next==head)
    {
      head=NULL;
      free(p);
      return(head);
    }
   for(q=head->next;q->next!=head;q=q->next)
   ;
   p=head;
   head=q;
   head->next=p->next;
   free(p);
   return(head);
}
node *delete_in(node *head)
{
  node *p,*q;
  int loc,i;
  if(head==NULL)
    {
            printf("\nUnderflow....Empty Linked List");
            return(head);
    }
  printf("\nEnter the location of the data to be deleted : ");
  scanf("%d",&loc);
  if(loc==1)
```

```
    {
      p=head->next;
      if(head->next==head)
            head=NULL;
      else
            head->next=p->next;
      free(p);
      return(head);
    }
   else
    {
          q=head->next;
          for(i=1; i<loc-1;i++)
                if(q->next->next!=head->next)
                                q=q->next;
                 else
                        {
                                printf("\nunderflow **** ");
                                return head;
                        }
          p=q->next;
          q->next=p->next;
          if(p==head)
            head=q;
          free(p);
    }
 return(head);
}
void search(node *head)
{ node *p;
  int data,loc=1;
  printf("\nEnter the data to be searched: ");
  scanf("%d",&data);
  p=head->next;
  do
   {
    if(data==p->data)
     {
          printf("\nFound at location=%d",loc);
          return;
     }
    loc++;
    p=p->next;
   }while(p!=head->next);
   printf("\nNot found:");
}
void print(node *head)
 {
```

```
        node *p;
        printf("\n\n");
        if(head==NULL)
                return;
        p=head->next;
        do
         {
                printf("%d  ",p->data);
                p=p->next;
         } while(p!=head->next);
}
```

OUTPUT:-