DEV KANSAL
IBM19CS046

```
#include <stdio.h>
#include <stdlib.h>
  struct node
  {
  int data;
  struct node *next;
  };
void qinsert();
void display();
void qdel();
void spush();
void spop();
void sdisplay();
struct node *rear=NULL, *front=NULL, *top=NULL;
int main(int argc, char** argv)
    {
     int choice;
     printf("Enter the choice \n1. stack \n2. queue\n");
  scanf("%d", &choice);
  if(choice == 1)
 do {
    printf("\n1. push\n2. display\n3. pop\n");
printf("Enter your choice");
  scanf("%d", &choice);
  switch(choice)
  {
  case 1 : spush(); break;
  case 2 : sdisplay; break;
  case 3 :  spop; break;
  default: if(choice != 4)
```

```
Printf("\n Invalid Input");
}} while(choice!=4);
else if(choice==2)
do { printf("\n Queue implementation using
        linked list\n");
Printf("\n1. Create \n2. Display \n3. Delete \n4. Exit \n");
Printf("\n Enter your choice");
scanf("%d", &choice);
switch(choice)
{ case 1: q.insert(); break;
  case 2: q.display(); break;
  case 3: q.del(); break;
default: if(choice!=4)
    Printf("\n Invalid input");
}} while(choice!=4);}
Void q.insert()
{ Struct node *new node;
  newnode = (Struct node *)malloc(sizeof(structnode));
  Printf("Enter the element:\n");
  scanf("%d", &newnode -> data);
  newnode -> next = NULL;
  if(rear = NULL)
```

```
}
  rear = newnode;
  front = newnode;
  }
  else
  { rear->next = newnode;
  rear = newnode;
  }}
  void q_del()
  { if (front == NULL)
  { printf ("Queue is empty\n"); return; }
  else { printf ("Deleted el is %d", &front->data);
    if (front == rear)
  { printf ("Queue is empty\n");
    front = NULL; rear = NULL;
  }
  else
  front = front->next; }}
   void q_display ()
   {
     struct node *temp;
  if (front == NULL)
   { printf ("Queue is empty ");
     return;
  }
```

```
temp= front;
while(temp!= NULL)
{
  printf("%d", temp->data);
  temp= temp-> next; }}
void spush()
{ int item;
Struct node * new node.
printf("Enter the element\n");
 Scanf("%d", &item);
new node = (Struct node *) malloc (sizeof (Struct node));
 new node->data=item;
new node ->next= NULL;
 if (top==NULL)
    top= new node;
    else
      new node -> next = top;
       top= new node;
  }
  void spop()
   { if (top==NULL)
      printf("Stack is empty");
else { printf("element removed is %d:", top->data);
   top= top -> next; }}
```

```
Void sdisp()
{
  Struct node *temp;
  temp=top;
  if(top==NULL)
  Printf("Stack is empty");
  while(temp!=NULL)
{Printf("%d", temp->data);
  temp=temp->next;
  }}
```