2)WAP to implement
Stack & Queues using Linked Representation

1. Stack

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct NODE{
    int data;
    struct NODE*link;
};
typedef struct NODE node;
node *top=NULL;
void push();
void pop();
void display();
void main()
{
    printf("1.INSERT\t 2.DELETE\t 3.DISPLAY\t 4.EXIT");
    int c;
    while(1)
    {
        printf("\nEnter your choice");
        scanf("%d",&c);
        switch(c)
        {
            case 1 :push();
```

```
                break;
            case 2 :pop();
                break;
            case 3 :display();
                break;
            case 4 :exit(0);
                break;
            default :printf("Invalid input");
        }
    }
}
void push(){
    node *new;
    new= (node*)malloc(sizeof(node));
    printf("\nEnter Element\n");
    scanf("%d",&new->data);
    if(top==NULL){
        top=new;
        top->link=NULL;
        return;
    }
    new->link=top;
    top=new;
}
void pop(){
    node *temp;
```

```
    if(top==NULL){
        printf("stack is empty");
        return;
    }
    temp=top;
    top=top->link;
    printf("deleted element is :%d\n",temp->data);
    free(temp);
}
void display(){
    node *temp;
    if(top==NULL){
        printf("stack is empty\n");
        return;
    }
    printf("Elements are:\n");
    temp=top;
    while(temp!=NULL){
        printf("%d\t",temp->data);
        temp=temp->link;
    }
}
```

2. Queue

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
```

```
#include<malloc.h>
struct NODE{
   int data;
   struct NODE*link;
};
typedef struct NODE node;
node *front=NULL;
node *rear=NULL;
void main()
{
   printf("1.INSERT\t 2.DELETE\t 3.DISPLAY\t 4.EXIT\n");
   int c;
   while(1)
   {
      printf("\nEnter your choice\n");
      scanf("%d",&c);
      switch(c)
      {
         case 1 :insert();
                break;
         case 2 :delete();
                break;
         case 3 :display();
                break;
         case 4 :exit(0);
```

```
                break;
        default :printf("\nInvalid input\n");
      }
   }
}
void insert(){
   node *new;
   new=(node*)malloc(sizeof(node));
   printf("\nEnter element\n");
   scanf("%d",&new->data);
   if(front==NULL && rear==NULL){
      rear=new;
      rear->link=NULL;
      front=rear;
      return;
   }
   rear->link=new;
   rear=new;
   rear->link=NULL;
}
void delete(){
   if(front==NULL){
      printf("\nQueue is empty\n");
      return;
   }
```

```
    node *temp;
    temp=front;
    printf("\ndeleted elements is:%d\n",temp->data);
    if(front==rear){
        front=NULL;
        rear=NULL;
    }
    else
    front=front->link;
    free(temp);
}
void display(){
    if(front==NULL){
        printf("Queue is empty\n");
        return;
    }
    node *temp;
    temp=front;
    printf("The elements are:\n");
    while(temp!=NULL){
        printf("%d\t",temp->data);
        temp=temp->link;
    }
}
```