8) WAP to implement stack & queues using linked representation.

```
// stack implementation

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

void push();
void pop();
void display();

struct node {
    int data;
    struct node *next;
};

struct node *top=NULL;

int main(){
    int choice;
    char ch;
    do{
    printf("\n1-Push \n2-Display
            \n3-Pop\n");
    printf("\nEnter your choice:");
    scanf("%d",&choice);

    switch(choice){
    case 1: push();
            break();

    case 2: display();
            break;
```

```
        case 3 : pop();
                 break;
    }
    printf("\n Do you want to continue
                (y||Y) : ");
    fflush(stdin);
    scanf("%c", &ch);
  } while (ch=='y' || ch=='Y');
}

void push(){

    int item;
    struct node *newnode;
    printf("Enter the element\n");
    scanf("%d", &item);

    newnode = (struct node *) malloc
                (sizeof(struct node));
    newnode->data = item;
    newnode->next = NULL;

    if (top==NULL)
        top=newnode;
    else
        newnode->next = top;
        top = newnode;
}

void pop(){

    if (top==NULL)
        printf("stack is empty");
    else {
```

```
        printf("Element removed is %d:",
                top->data);
        top = top->next;
    }
}

void display(){

    struct node *temp;
    temp = top;

    if (top=NULL)
      printf("stack is empty");

    while (temp!=NULL){
      printf("%d", temp->data);
      temp = temp->next;
    }
}

// queue implementation

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

struct node {
   int data;
   struct node *next;
};

void insert();
void display();
void del();
```

```
struct node *rear=NULL, *front=NULL;

int main() {

    int choice;
    char ch='Y';
    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: insert();
            break;
    case 2: display();
            break;
    case 3: del();
            break;
    case 4: ch='n';
            break;
    }
    } while(ch=='y'||ch=='Y');
}

void insert() {
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof
            (struct node));
    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 del() {

    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
    else {
        printf("Deleted ele is %d",
                front->data);
        if (front == rear) {
            printf("Queue is empty\n");
            front = NULL;
            rear = NULL;
        }
        else {
            front = front->next;
        }
    }
}

void display() {
    struct node *temp;
    if (front == NULL) {
        printf("Queue is empty\n");
```

```
    return;
  }
  temp = front;
  while ( temp != NULL) {
    printf ("%d", temp->data);
    temp = temp->next;
  }
}
```