8. Write a program to implement Stack & Queues using Linked Representation

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

struct node
{
   int info;
   struct node *ptr;
}*top,*top1,*temp,*front,*rear,*front1;

int topelement();
void push(int data);
void pop();
void empty_stack();
void display_stack();
void destroy();
void stack_count();
void create_stack();
int frontelement();
void enq(int data);
void deq();
void empty_queue();
void display_queue();
```

```
void create_queue();
void queuesize();

int count = 0;

void main()
{
   int no, ch, e;

   printf("\n 1 - Push");
   printf("\n 2 - Pop");
   printf("\n 3 - Top");
   printf("\n 4 - Empty the stack");
   printf("\n 5 - Exit");
   printf("\n 6 - Display the stack");
   printf("\n 7 - Stack Count");
   printf("\n 8 - Destroy stack");
   printf("\n 9 - Enque");
   printf("\n 10 - Deque");
   printf("\n 11 - Front element");
   printf("\n 12 - Empty the queue");
   printf("\n 13 - Display the queue");
   printf("\n 14 - Queue size");
   create_stack();
```

```
create_queue();

while (1)
{
   printf("\n Enter choice : ");
   scanf("%d", &ch);

   switch (ch)
   {
   case 1:
      printf("Enter data : ");
      scanf("%d", &no);
      push(no);
      break;
   case 2:
      pop();
      break;
   case 3:
      if (top == NULL)
         printf("No elements in stack");
      else
      {
         e = topelement();
         printf("\n Top element : %d", e);
```

```
    }
    break;
case 4:
    empty_stack();
    break;
case 5:
    exit(0);
case 6:
    display_stack();
    break;
case 7:
    stack_count();
    break;
case 8:
    destroy();
    break;
case 9:
    printf("Enter data : ");
    scanf("%d", &no);
    enq(no);
    break;
case 10:
    deq();
    break;
```

```
        case 11:
            e = frontelement();
            if (e != 0)
                printf("Front element : %d", e);
            else
                printf("\n No front element in Queue as queue is empty");
            break;
        case 12:
            empty_queue();
            break;
        case 13:
            display_queue();
            break;
        case 14:
            queuesize();
            break;
        default :
            printf(" Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

void create_stack()
```

```
{
   top = NULL;
}

void stack_count()
{
   printf("\n No. of elements in stack : %d", count);
}

void push(int data)
{
   if (top == NULL)
   {
      top =(struct node *)malloc(1*sizeof(struct node));
      top->ptr = NULL;
      top->info = data;
   }
   else
   {
      temp =(struct node *)malloc(1*sizeof(struct node));
      temp->ptr = top;
      temp->info = data;
      top = temp;
   }
```

```
    count++;
}

void display_stack()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("Stack is empty");
        return;
    }

    while (top1 != NULL)
    {
        printf("%d ", top1->info);
        top1 = top1->ptr;
    }
 }

void pop()
{
    top1 = top;
```

```
    if (top1 == NULL)
    {
        printf("\n Error : Trying to pop from empty stack");
        return;
    }
    else
        top1 = top1->ptr;
    printf("\n Popped value : %d", top->info);
    free(top);
    top = top1;
    count--;
}

int topelement()
{
    return(top->info);
}

void empty_stack()
{
    if (top == NULL)
        printf("\n Stack is empty");
    else
        printf("\n Stack is not empty with %d elements", count);
```

```
}

void destroy()
{
   top1 = top;

   while (top1 != NULL)
   {
      top1 = top->ptr;
      free(top);
      top = top1;
      top1 = top1->ptr;
   }
   free(top1);
   top = NULL;

   printf("\n All stack elements destroyed");
   count = 0;
}
void create_queue()
{
   front = rear = NULL;
}
```

```
void queuesize()
{
   printf("\n Queue size : %d", count);
}

void enq(int data)
{
   if (rear == NULL)
   {
      rear = (struct node *)malloc(1*sizeof(struct node));
      rear->ptr = NULL;
      rear->info = data;
      front = rear;
   }
   else
   {
      temp=(struct node *)malloc(1*sizeof(struct node));
      rear->ptr = temp;
      temp->info = data;
      temp->ptr = NULL;

      rear = temp;
   }
   count++;
```

```
}

void display_queue()
{
   front1 = front;

   if ((front1 == NULL) && (rear == NULL))
   {
      printf("Queue is empty");
      return;
   }
   while (front1 != rear)
   {
      printf("%d ", front1->info);
      front1 = front1->ptr;
   }
   if (front1 == rear)
      printf("%d", front1->info);
}

void deq()
{
   front1 = front;
```

```
    if (front1 == NULL)
    {
        printf("\n Error: Trying to display elements from empty
queue");
        return;
    }
    else
        if (front1->ptr != NULL)
        {
            front1 = front1->ptr;
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = front1;
        }
        else
        {
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = NULL;
            rear = NULL;
        }
        count--;
}
```

```
int frontelement()
{
   if ((front != NULL) && (rear != NULL))
      return(front->info);
   else
      return 0;
}

void empty_queue()
{
    if ((front == NULL) && (rear == NULL))
      printf("\n Queue empty");
   else
     printf("Queue not empty");
}
```

Output:

```
 1 - Push
 2 - Pop
 3 - Top
 4 - Empty the stack
 5 - Exit
 6 - Display the stack
 7 - Stack Count
 8 - Destroy stack
 9 - Enque
 10 - Deque
 11 - Front element
 12 - Empty the queue
 13 - Display the queue
 14 - Queue size
 Enter choice : 1
Enter data : 56

 Enter choice : 1
Enter data : 80

 Enter choice : 2

 Popped value : 80
 Enter choice : 3

 Top element : 56
 Enter choice : 1
Enter data : 78

 Enter choice : 1
Enter data : 90

 Enter choice : 6
90 78 56
```

```
 Enter choice : 7

 No. of elements in stack : 3
 Enter choice : 8

 All stack elements destroyed
 Enter choice : 4

 Stack is empty
 Enter choice : 9
Enter data : 14

 Enter choice : 9
Enter data : 85

 Enter choice : 9
Enter data : 36

 Enter choice : 11
Front element : 14
 Enter choice : 13
14 85 36
 Enter choice : 14

 Queue size : 3
 Enter choice : 10

 Dequed value : 14
 Enter choice : 13
85 36
 Enter choice : 14

 Queue size : 2
 Enter choice : 12
Queue not empty
 Enter choice : 5
```