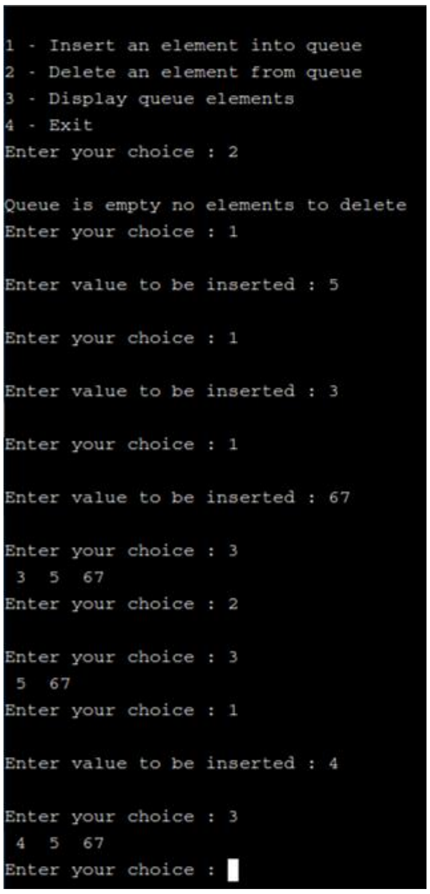
Lab 6

Q1

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

#define MAX 5

void pqinsert(int);
void pqmindelete(int);
void create();
void check(int);
void pqdisplay();

int pri_que[MAX];
int front, rear;

void main()
{
    int n, ch;
    printf("\n1 - Insert an element into queue");
    printf("\n2 - Delete an element from queue");
    printf("\n3 - Display queue elements");
    printf("\n4 - Exit");
    create();
    while (1)
    {
        printf("\nEnter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("\nEnter value to be inserted : ");
            scanf("%d",&n);
            pqinsert(n);
            break;
        case 2:
            pqmindelete(n);
            break;
        case 3:
            pqdisplay();
            break;
        case 4:
            exit(0);
        default:
            printf("\nChoice is incorrect, Enter a correct choice");
        }
    }
}
```

```c
void create()
{
    front = rear = -1;
}

void pqinsert(int data)
{
    if (rear >= MAX - 1)
    {
        printf("\nQueue overflow no more elements can be inserted");
        return;
    }
    if ((front == -1) && (rear == -1))
    {
        front++;
        rear++;
        pri_que[rear] = data;
        return;
    }
    else
        check(data);
    rear++;
}

void check(int data)
{
    int i,j;
    for (i = 0; i <= rear; i++)
    {
        if (data <= pri_que[i])
        {
            for (j = rear + 1; j > i; j--)
            {
                pri_que[j] = pri_que[j - 1];
            }
            pri_que[i] = data;
            return;
        }
    }
    pri_que[i] = data;
}

void pqmindelete(int data)
{   int i;
    if ((front==-1) && (rear==-1))
    {
        printf("\nQueue is empty no elements to delete");
        return;
    }
    for (i = 0; i < rear; i++)
        {
            pri_que[i] = pri_que[i + 1];
        }
    pri_que[i] = -99;
    rear--;
    if (rear == -1)
        front = -1;
    return;
}

void pqdisplay()
{
    if ((front == -1) && (rear == -1))
    {
        printf("\nQueue is empty");
        return;
    }
    for (; front <= rear; front++)
    {
        printf(" %d ", pri_que[front]);
    }
    front = 0;
}
```

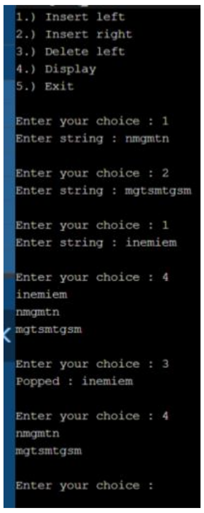


```
1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
Enter your choice : 2

Queue is empty no elements to delete
Enter your choice : 1

Enter value to be inserted : 5

Enter your choice : 1

Enter value to be inserted : 3

Enter your choice : 1

Enter value to be inserted : 67

Enter your choice : 3
 3  5  67
Enter your choice : 2

Enter your choice : 3
 5  67
Enter your choice : 1

Enter value to be inserted : 4

Enter your choice : 3
 4  5  67
Enter your choice : 
```

Q2

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 5
#define MAX_STR 10
typedef struct
{
   char arr[MAX_SIZE][MAX_STR];
   int front,rear;

}DQ_STR;

void init(DQ_STR *s)
{
   s->front = s->rear = -1;
}

int isEmpty(DQ_STR *s)
{
   if(s->rear == -1)
      return 1;
   return 0;
}

int isFull(DQ_STR *s)
{
   if((s->rear+1)%MAX_SIZE == s->front)
      return 1;
   return 0;
}

void insertright(DQ_STR *s, char x[])
{
   int i;
   if(isEmpty(s))
   {
      s->rear = s->front =0;
      for(i=0;x[i]!='\0';i++)
         s->arr[s->rear][i] = x[i];
      s->arr[s->rear][i] = '\0';
   }
   else
   {
      s->rear = (s->rear+1)%MAX_SIZE;
      for(i=0;x[i]!='\0';i++)
         s->arr[s->rear][i] = x[i];
      s->arr[s->rear][i] = '\0';
   }
}
void insertleft(DQ_STR *s, char x[])
{
   int i;
```

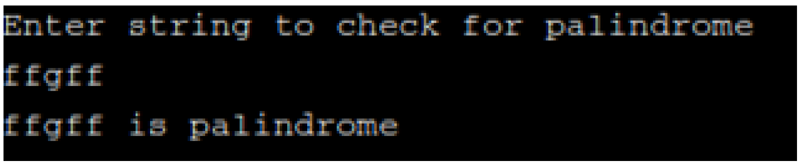
```
    if(isEmpty(s))
    {
        s->rear = s->front =0;
        for(i=0;x[i]!='\0';i++)
            s->arr[s->front][i] = x[i];
        s->arr[s->front][i] = '\0';
    }
    else
    {
        s->front = (s->front-1+MAX_SIZE)%MAX_SIZE;
        for(i=0;x[i]!='\0';i++)
            s->arr[s->front][i] = x[i];
        s->arr[s->front][i] = '\0';

    }
}

char* deleteleft(DQ_STR *s)
{
    char *str;
    str = s->arr[s->front];
    if(s->rear == s->front)
    { init(s); }
    else
    { s->front = (s->front+1)%MAX_SIZE; }
    return str;
}
void displaydq(DQ_STR *s)
{
    if(isEmpty(s))
    {
        printf("Queue is empty\n");
        return;
    }
    for(int temp = (s->front)%MAX_SIZE; temp!=(s->rear); temp=(temp+1)%MAX_SIZE)
        printf("%s\n",s->arr[temp]);
    printf("%s\n",s->arr[s->rear]);
}

int main()
{
    DQ_STR s;
    init(&s);
    int ch;
    char str[MAX_STR];
    printf("1.) Insert left\n2.) Insert right\n3.) Delete left\n4.) Display\n5.) Exit\n");
    while(1)
    {
        printf("\nEnter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
         case 1:
             if(isFull(&s))
                 printf("Overflow\n");
             else
             {
                 printf("Enter string : ");
                 scanf("%s",str);
                 insertleft(&s,str);
             }
             break;
          case 2:
             if(isFull(&s))
                 printf("Overflow\n");
             else
             {
                 printf("Enter string : ");
                 scanf(" %s",str);
                 insertright(&s,str);
             }
             break;
          case 3 :
             if(!isEmpty(&s))
             {
                 char *pop = deleteleft(&s);
                 printf("Popped : %s\n",pop);
             }
             else
                 printf("Underflow\n");
             break;
          case 4 :
             displaydq(&s);
             break;
          case 5 :
             exit (0);
          default :
             printf("Wrong number! Try Again");
        }
    }
}
```



```
1.) Insert left
2.) Insert right
3.) Delete left
4.) Display
5.) Exit

Enter your choice : 1
Enter string : nmgmtn

Enter your choice : 2
Enter string : mgtsmtgsm

Enter your choice : 1
Enter string : inemiem

Enter your choice : 4
inemiem
nmgmtn
mgtsmtgsm

Enter your choice : 3
Popped : inemiem

Enter your choice : 4
nmgmtn
mgtsmtgsm

Enter your choice :
```

Q3
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 30

typedef struct dequeue
{
   char data[MAX];
   int rear,front;
}dequeue;

void initialize(dequeue *P)
{
   P->rear=-1;
   P->front=-1;
}
int empty(dequeue *P)
{
   if(P->rear==-1)
   return(1);
   return(0);}
   int full(dequeue *P)
   {
   if((P->rear+1)%MAX==P->front)
   return(1);
   return(0);
}
void enqueueR(dequeue *P,char x)
{
   if(empty(P))
   {
      P->rear=0;
      P->front=0;
      P->data[0]=x;
   }
   else
   {
      P->rear=(P->rear+1)%MAX;
      P->data[P->rear]=x;
   }
}
void enqueueF(dequeue *P,char x)
{
   if(empty(P))
   {
      P->rear=0;
      P->front=0;
      P->data[0]=x;
      }
```

```
        else{
        P->front=(P->front-1+MAX)%MAX;
        P->data[P->front]=x;
    }
}

char dequeueF(dequeue *P)
{
    char x;
    x=P->data[P->front];
    if(P->rear==P->front)
    /*delete the last element */
    initialize(P);
    else
    P->front=(P->front+1)%MAX;
    return(x);
}
char dequeueR(dequeue *P)
{
    char x;
    x=P->data[P->rear];
    if(P->rear==P->front)
    initialize(P);
    else
    P->rear=(P->rear-1+MAX)%MAX;
    return(x);
}
void print(dequeue *P)
{
    if(empty(P))
    {
        printf("\nQueue is empty!!");exit(0);
    }
    int i;
    i=P->front;
    while(i!=P->rear)
    {
        printf("\n%c",P->data[i]);



        i=(i+1)%MAX;
    }
    printf("\n%c\n",P->data[P->rear]);
}

int main()
{
    int i,x,n;
    int op=0;
    char c[20];
    dequeue q;
    initialize(&q);
    printf("Enter string to check for palindrome\n");
    scanf("%s",c);
    n= strlen(c);
    for(i=0;i<n;i++)
    {
        enqueueF(&q,c[i]);
    }
    for(i=0;i<n/2;i++)
    {
        if(dequeueF(&q)!=dequeueR(&q))
        {
            op = 1;
            break;
        }
    }
    if(op == 0)
        printf("%s is palindrome\n",c);
    else
        printf("%s is not palindrome\n",c);
    return 0;
}
```



```
Enter string to check for palindrome
ffgff
ffgff is palindrome
```