Q1:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5

typedef struct{
	int a[MAX];
	int front,rear;
}APQ;

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

void pqinsert(APQ* apq,int data)
{
	if (apq->rear >= MAX - 1)
	{
		printf("\nQueue overflow no more elements can be inserted");
		return;
	}
	if ((apq->front == -1) && (apq->rear == -1))
	{
		apq->front++;
		apq->rear++;
		apq->a[apq->rear] = data;
		return;
	}
	else
		check(apq,data);
	apq->rear++;
}
void pqmindelete(APQ* apq)
{
	int i;
	if ((apq->front==-1) && (apq->rear==-1))
	{
		printf("\nQueue is empty no elements to delete");
```

```
                return;
        }
        for (i = 0; i < apq->rear; i++)
        {
                apq->a[i] = apq->a[i + 1];}
                apq->a[i] = -99;
                apq->rear--;
                if (apq->rear == -1)
                        apq->front = -1;
                return;
        }


void pqdisplay(APQ* apq)
{
        if ((apq->front == -1) && (apq->rear == -1))
        {
                printf("\nQueue is empty");
                return;
        }
        for (; apq->front <= apq->rear; apq->front++)
        {
                printf(" %d ", apq->a[apq->front]);
        }
        apq->front = 0;
}
void main()
{
        APQ apq;
        apq.front=-1;
        apq.rear=-1;

        int n, ch,flag=1;
        while (flag)
        {
                printf("\n1 - Insert an element into queue");
                printf("\n2 - Delete an element from queue");
                printf("\n3 - Display queue elements");
                printf("\n4 - Exit");
                printf("\nEnter your choice : ");
                scanf("%d", &ch);
                switch (ch)
                {
                        case 1:
                        printf("\nEnter value to be inserted : ");
                        scanf("%d",&n);
                        pqinsert(&apq,n);
                        break;
                        case 2:
                        pqmindelete(&apq);
                        break;
                        case 3:
```

```
                pqdisplay(&apq);
                break;
                case 4:
                flag=0;
                default:
                printf("\nChoice is incorrect, Enter a correct choice");
            }
        }
        return ;
}
```

Output:

```
student@dslab:~/190905216-DSA/Programs/queueappli$ ./a.out

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

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

Enter value to be inserted : 78

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

Enter value to be inserted : 32

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

Enter value to be inserted : 7

1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
Enter your choice : 3
 7  32  78
1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
```

```
Enter your choice : 2

1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
Enter your choice : 3
 32  78
1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
Enter your choice : 4

student@dslab:~/190905216-DSA/Programs/queueappli$
```

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");
		}
	}
}
```

Output:

```
student@dslab:~/190905216-DSA/Programs/queueappli$ gcc QueueStrings.c
student@dslab:~/190905216-DSA/Programs/queueappli$ ./a.out
1.) Insert left
2.) Insert right
3.) Delete left
4.) Display
5.) Exit

Enter your choice : 1
Enter string : Hello

Enter your choice : 2
Enter string : Spanish

Enter your choice : 1
Enter string : Data

Enter your choice : 4
Data
Hello
Spanish

Enter your choice : 3
Popped : Data

Enter your choice : 4
Hello
Spanish

Enter your choice : 5
student@dslab:~/190905216-DSA/Programs/queueappli$
```

Q3:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 30
typedef struct DQ
{
        char data[MAX];
        int rear,front;
}DQ;


int empty(DQ *Q)
{
        if(Q->rear==-1)
                return(1);
        return(0);}
        int full(DQ *Q)
        {
                if((Q->rear+1)%MAX==Q->front)
                        return(1);
                return(0);
        }
```

```
void enqueueR(DQ *Q,char x)
{
	if(empty(Q))
	{
		Q->rear=0;
		Q->front=0;
		Q->data[0]=x;
	}
	else
	{
		Q->rear=(Q->rear+1)%MAX;
		Q->data[Q->rear]=x;
	}
}
void enqueueF(DQ *Q,char x)
{
	if(empty(Q))
	{
		Q->rear=0;
		Q->front=0;
		Q->data[0]=x;
	}else{
		Q->front=(Q->front-1+MAX)%MAX;
		Q->data[Q->front]=x;
	}
}
char dequeueF(DQ *Q)
{
	char x;
	x=Q->data[Q->front];
	if(Q->rear==Q->front){
		Q->rear=-1;Q->front=-1;
	}
	else
		Q->front=(Q->front+1)%MAX;
	return(x);
}
char dequeueR(DQ *Q)
{
	char x;
	x=Q->data[Q->rear];
	if(Q->rear==Q->front){
		Q->rear=-1;Q->front=-1;
	}
	else
		Q->rear=(Q->rear-1+MAX)%MAX;
	return(x);
}
void print(DQ *Q)
{
	if(empty(Q))
	{ main
```

```
                printf("\nQueue is empty!!");exit(0);
        }
        int i;
        i=Q->front;
        while(i!=Q->rear)
        {
                printf("\n%c",Q->data[i]);
                i=(i+1)%MAX;
        }
        printf("\n%c\n",Q->data[Q->rear]);
}
int main()
{
        int i,x,n;
        int op=0;
        char c[20];

        DQ Q;
        Q.rear=-1;
        Q.front=-1;

        printf("Enter string to check for Palindrome:  ");
        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 a Palindrome\n",c);
        return 0;
}
```

Output:

```
student@dslab:~/190905216-DSA/Programs/queueappli$ gcc PallindromeQueue.c
student@dslab:~/190905216-DSA/Programs/queueappli$ ./a.out
Enter string to check for palindrome:  malayalam
malayalam is palindrome
student@dslab:~/190905216-DSA/Programs/queueappli$ ./a.out
Enter string to check for palindrome:  google
google is not palindrome
```