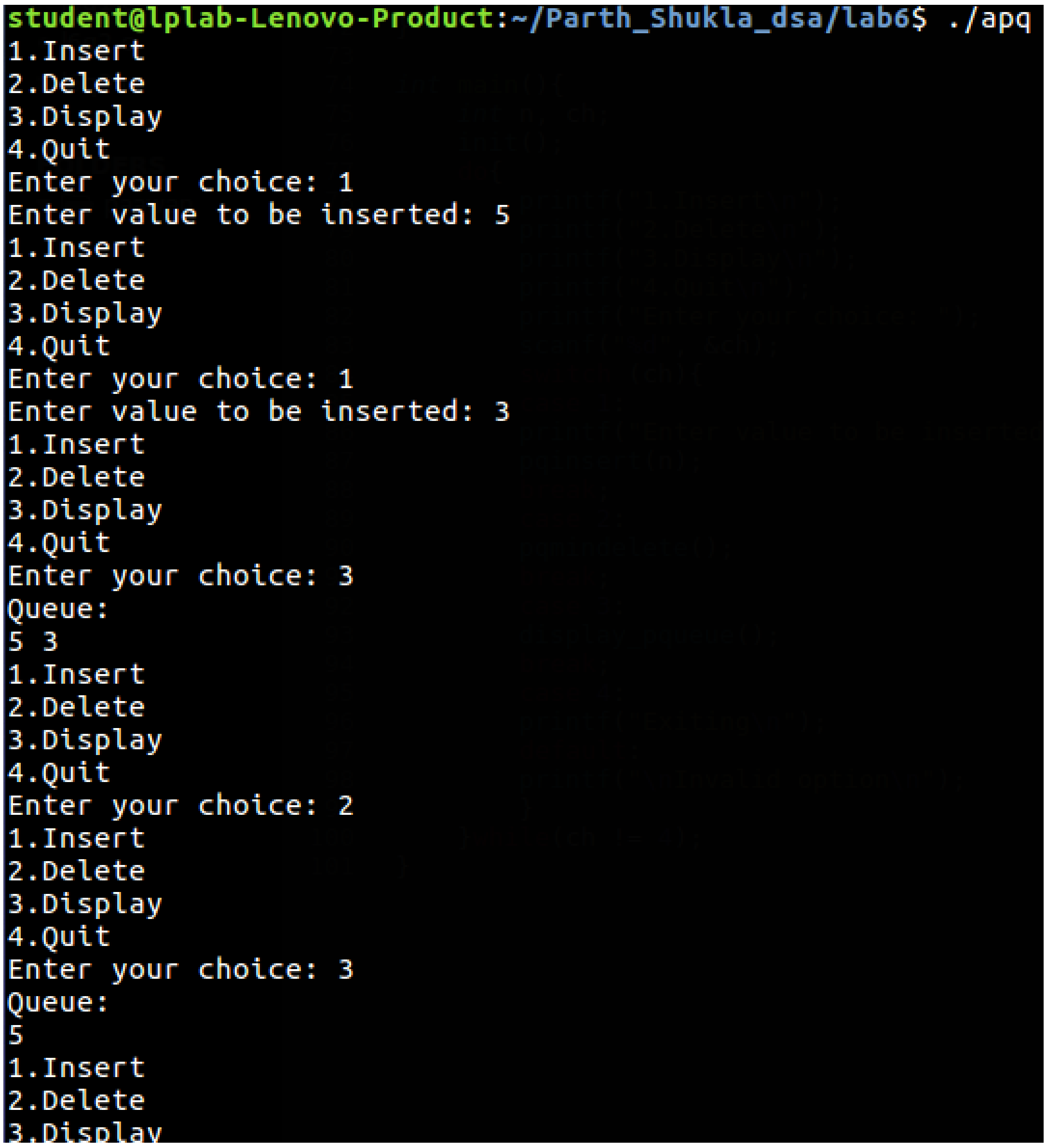
Week 6

1)

```
// Implement an ascending priority queue.

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define MAX 10

typedef struct{
	int pri_que[MAX];
	int front, rear;
}pq;

void init(pq* q){
	q->front = -1;
	q->rear = -1;
}

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

void pqinsert(pq* q, int data)
{
	if (q->rear >= MAX - 1){
		printf("Queue overflow\n");
		return;
	}
	if((q->front == -1) && (q->rear == -1)){
		q->front++;
		q->rear++;
		q->pri_que[q->rear] = data;
		return;
	}
	else
		check(q, data);
	q->rear++;
}
```

```
void pqmindelete(pq* q){
	int i;
	if ((q->front==-1) && (q->rear==-1)){
		printf("\nQueue Underflow");
		return;
	}
	q->rear = q->rear - 1;
}

void display_pqueue(pq* q){
	if ((q->front == -1) && (q->rear == -1))
	{
		printf("\nQueue is empty");
		return;
	}
	printf("Queue:\n");
	for (; q->front <= q->rear; q->front++)
	{
		printf("%d ", q->pri_que[q->front]);
	}
	printf("\n");
	q->front = 0;
}

int main(){
	pq pri;
	pq* q = &pri;
	int n, ch;
	init(q);
	do{
		printf("1.Insert\n");
		printf("2.Delete\n");
		printf("3.Display\n");
		printf("4.Quit\n");
		printf("Enter your choice: ");
		scanf("%d", &ch);
		switch (ch){
		case 1:
		printf("Enter value to be inserted: ");
		scanf("%d",&n);
		pqinsert(q, n);
		break;
		case 2:
		pqmindelete(q);
		break;
		case 3:
		display_pqueue(q);
		break;
		case 4:
		printf("Exiting\n");
```

```
                break;
                default:
                printf("\nInvalid option\n");
                }
        }while(ch != 4);
}
```



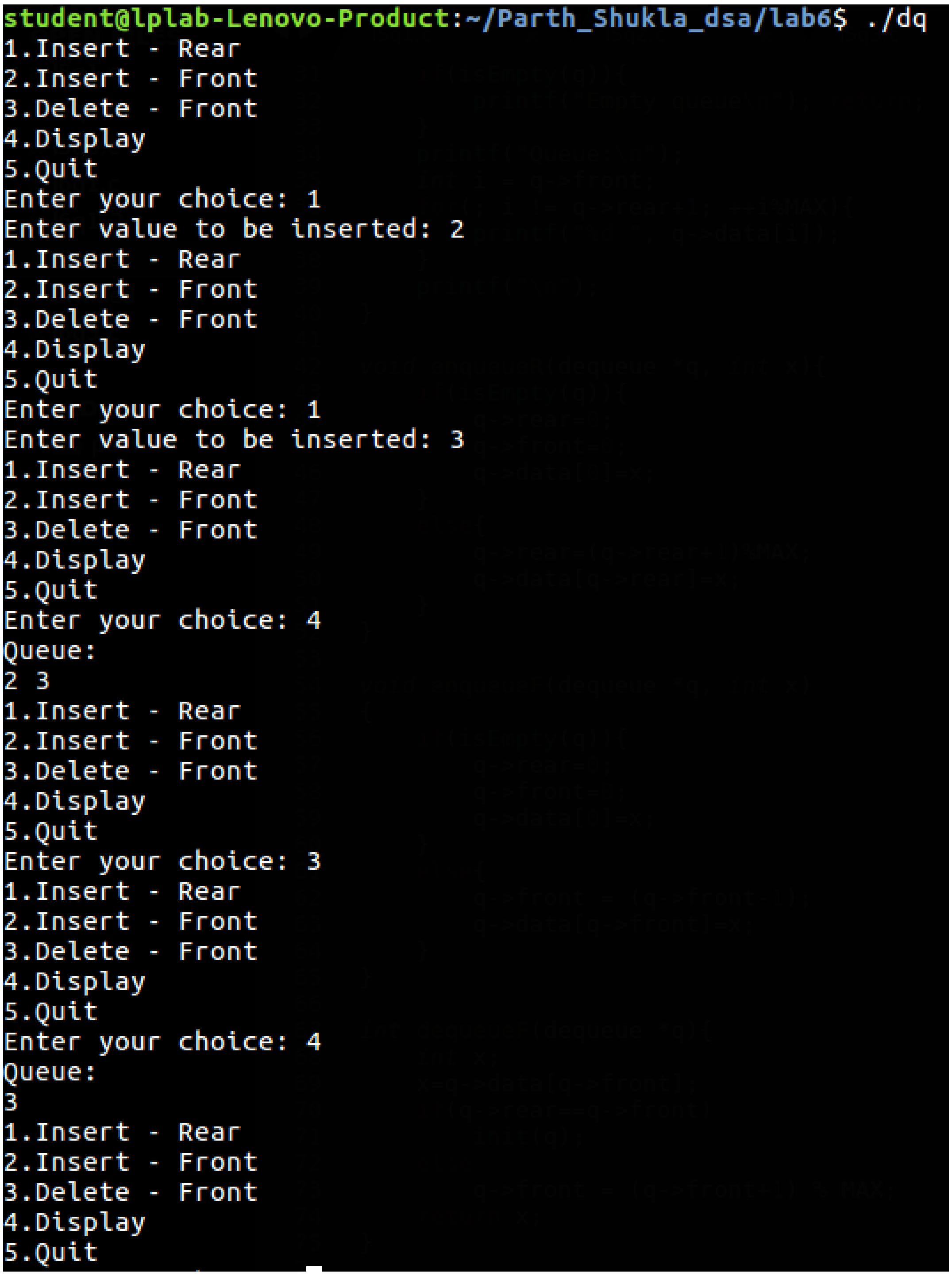
```
student@lplab-Lenovo-Product:~/Parth_Shukla_dsa/lab6$ ./apq
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice: 1
Enter value to be inserted: 5
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice: 1
Enter value to be inserted: 3
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice: 3
Queue:
5 3
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice: 2
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice: 3
Queue:
5
1.Insert
2.Delete
3.Display
```

2)

```
// Implement a queue of strings using an output restricted dequeue (no   deleteRight).

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define MAX 10

typedef struct{
        int data[MAX];
        int front, rear;
```

```
}dequeue;

void init(dequeue *dq){
        dq->front = -1;
        dq->rear = -1;
}

int isEmpty(dequeue *dq){
        if(dq->front == -1 && dq->rear == -1)
                return 1;
        return 0;
}

int isFull(dequeue *dq){
        if((dq->rear+1) % MAX == dq->front)
                return 1;
        return 0;
}

void display(dequeue *q){
        if(isEmpty(q)){
                printf("Empty queue\n"); return;
        }
        printf("Queue:\n");
        int i = q->front;
        for(; i != q->rear+1; ++i%MAX){
                printf("%d ", q->data[i]);
        }
        printf("\n");
}

void enqueueR(dequeue *q, int x){
        if(isEmpty(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(dequeue *q, int x)
{
        if(isEmpty(q)){
                q->rear=0;
                q->front=0;
                q->data[0]=x;
        }
        else{
                q->front = (q->front-1);
```

```
		q->data[q->front]=x;
	}
}

int dequeueF(dequeue *q){
	int x;
	x=q->data[q->front];
	if(q->rear==q->front)
		init(q);
	else
		q->front = (q->front+1) % MAX;
	return x;
}

int main(){
	dequeue dq;
	dequeue* q = &dq;
	int n, ch;
	init(q);
	do{
		printf("1.Insert - Rear\n");
		printf("2.Insert - Front\n");
		printf("3.Delete - Front\n");
		printf("4.Display\n");
		printf("5.Quit\n");
		printf("Enter your choice: ");
		scanf("%d", &ch);
		switch (ch){
		case 1:
		printf("Enter value to be inserted: ");
		scanf("%d",&n);
		enqueueR(q, n);
		break;
		case 2:
		printf("Enter value to be inserted: ");
		scanf("%d",&n);
		enqueueF(q, n);
		break;
		case 3:
		dequeueF(q);
		break;
		case 4:
		display(q);
		break;
		case 5:
		printf("Exiting\n");
		break;
		default:
		printf("\nInvalid option\n");
		}
	}while(ch != 5);
}
```

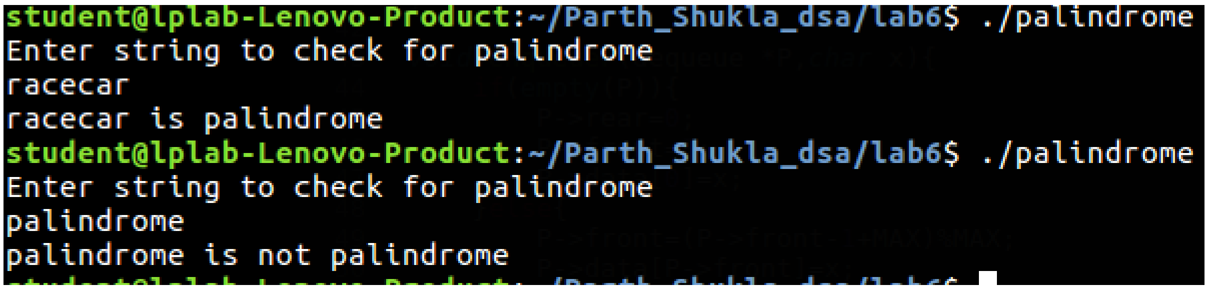


```
student@lplab-Lenovo-Product:~/Parth_Shukla_dsa/lab6$ ./dq
1.Insert - Rear
2.Insert - Front
3.Delete - Front
4.Display
5.Quit
Enter your choice: 1
Enter value to be inserted: 2
1.Insert - Rear
2.Insert - Front
3.Delete - Front
4.Display
5.Quit
Enter your choice: 1
Enter value to be inserted: 3
1.Insert - Rear
2.Insert - Front
3.Delete - Front
4.Display
5.Quit
Enter your choice: 4
Queue:
2 3
1.Insert - Rear
2.Insert - Front
3.Delete - Front
4.Display
5.Quit
Enter your choice: 3
1.Insert - Rear
2.Insert - Front
3.Delete - Front
4.Display
5.Quit
Enter your choice: 4
Queue:
3
1.Insert - Rear
2.Insert - Front
3.Delete - Front
4.Display
5.Quit
```

3)

```
// Write a program to check whether given string is a palindrome using a dequeue.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 30

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

void init(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)
                init(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)
		init(P);
	else
		P->rear=(P->rear-1+MAX)%MAX;
	return x;
}

void print(dequeue *P){
	if(empty(P))
	{
		printf("Queue 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 ans=0;
	char c[20];
	dequeue q;
	init(&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))
		{
			ans = 1;
			break;
		}
	}
	if(ans == 0)
		printf("%s is palindrome\n",c);
	else
		printf("%s is not palindrome\n",c);
	return 0;
}
```



```
student@lplab-Lenovo-Product:~/Parth_Shukla_dsa/lab6$ ./palindrome
Enter string to check for palindrome
racecar
racecar is palindrome
student@lplab-Lenovo-Product:~/Parth_Shukla_dsa/lab6$ ./palindrome
Enter string to check for palindrome
palindrome
palindrome is not palindrome
student@lplab-Lenovo-Product:~/Parth_Shukla_dsa/lab6$
```