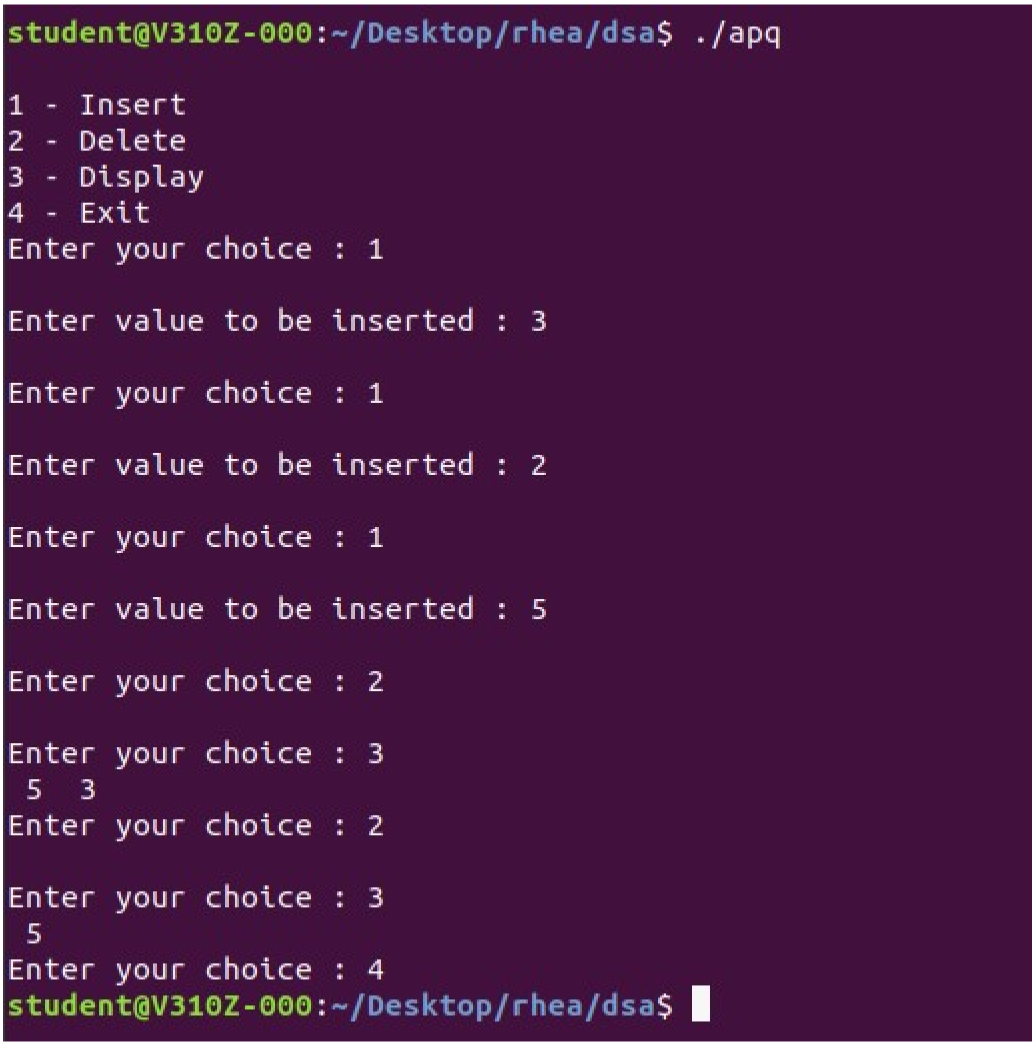
LAB 6

Q1 ASCENDING PRIORITY QUEUE

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

#define MAX 5

int pri_que[MAX];
int front, rear;

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

void pqinsert(int data)
{
   if (rear >= MAX - 1)
   {
      printf("\nQueue overflow");
      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 i;

    if ((front==-1) && (rear==-1))
    {
        printf("\nQueue Underflow");
        return;
    }

    rear = rear - 1;
}

void display_pqueue()
{
    if ((front == -1) && (rear == -1))
    {
        printf("\nQueue is empty");
        return;
    }

    for (; front <= rear; front++)
    {
        printf(" %d ", pri_que[front]);
    }

    front = 0;
}

void main()
{
    int n, ch;

    printf("\n1 - Insert");
    printf("\n2 - Delete");
    printf("\n3 - Display");
    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();
            break;
        case 3:
            display_pqueue();
            break;
        case 4:
            exit(0);
        default:
            printf("\nInvalid option\n");
        }
    }
}
```



```
student@V310Z-000:~/Desktop/rhea/dsa$ ./apq

1 - Insert
2 - Delete
3 - Display
4 - Exit
Enter your choice : 1

Enter value to be inserted : 3

Enter your choice : 1

Enter value to be inserted : 2

Enter your choice : 1

Enter value to be inserted : 5

Enter your choice : 2

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

Enter your choice : 3
 5
Enter your choice : 4
student@V310Z-000:~/Desktop/rhea/dsa$ 
```

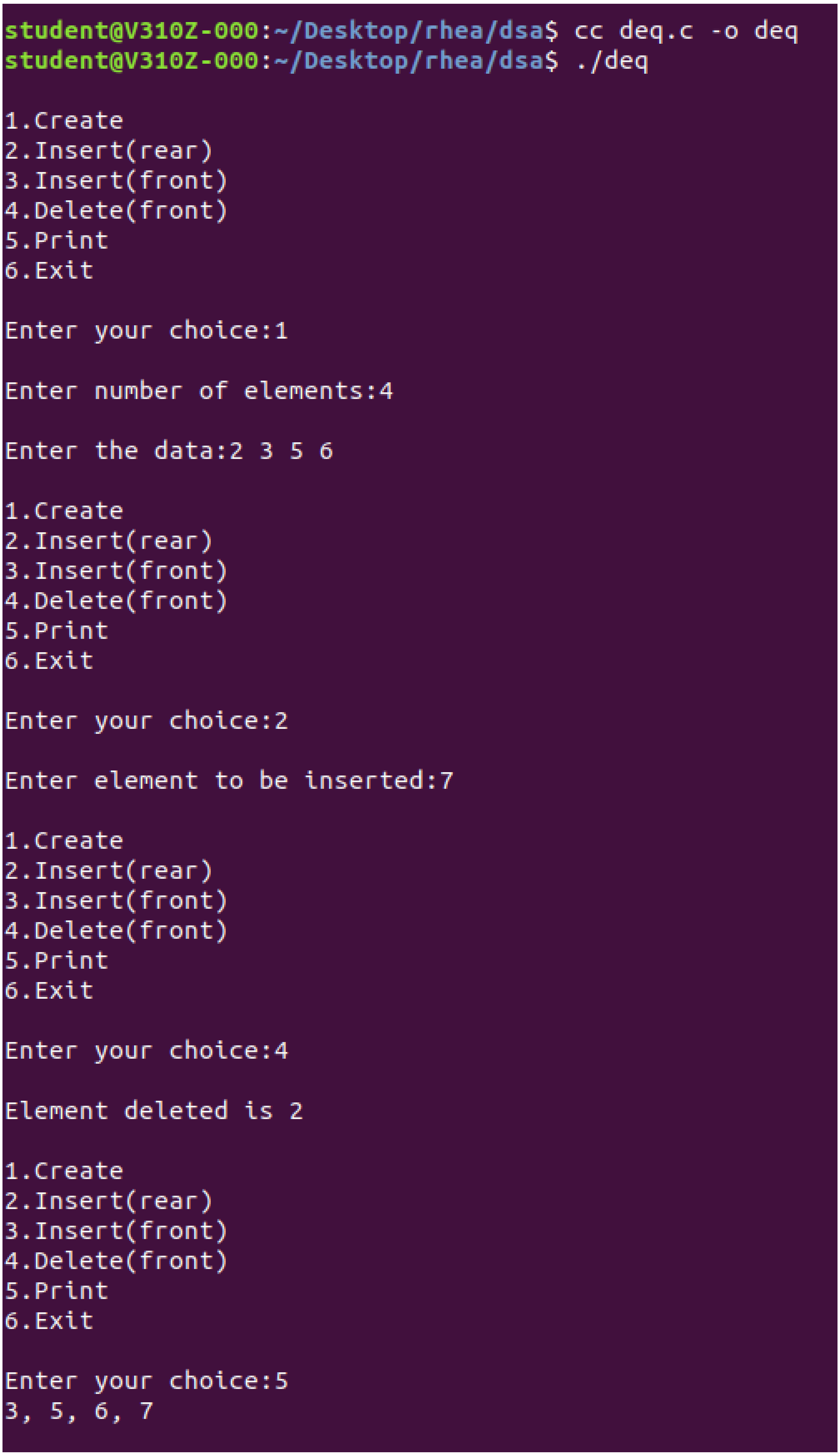
Q2 OUTPUT RSTRICTED DEQUEUE

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

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

void initialize(dequeue *p);
int empty(dequeue *p);
int full(dequeue *p);
void enqueueR(dequeue *p,int x);
void enqueueF(dequeue *p,int x);
int dequeueF(dequeue *p);
int dequeueR(dequeue *p);
void print(dequeue *p);

void main()
{
    int i,x,op,n;
    dequeue q;

    initialize(&q);

    do
    {
        printf("\n1.Create\n2.Insert(rear)\n3.Insert(front)\n4.Delete(front)");
        printf("\n5.Print\n6.Exit\n\nEnter your choice:");
        scanf("%d",&op);

        switch(op)
        {
            case 1: printf("\nEnter number of elements:");
                    scanf("%d",&n);
                    initialize(&q);
                    printf("\nEnter the data:");

                    for(i=0;i<n;i++)
                    {
                        scanf("%d",&x);
                        if(full(&q))
                        {
                            printf("\nQueue is full!!");
                            exit(0);
                        }
                        enqueueR(&q,x);
                    }
                    break;

            case 2: printf("\nEnter element to be inserted:");
                    scanf("%d",&x);

                    if(full(&q))
                    {
                        printf("\nQueue is full!!");
                        exit(0);
                    }

                    enqueueR(&q,x);
                    break;

            case 3: printf("\nEnter the element to be inserted:");
                    scanf("%d",&x);

                    if(full(&q))
                    {
                        printf("\nQueue is full!!");
                        exit(0);
                    }

                    enqueueF(&q,x);
                    break;

            case 4: if(empty(&q))
                    {
                        printf("\nQueue is empty!!");
                        exit(0);
                    }

                    x=dequeueF(&q);
                    printf("\nElement deleted is %d\n",x);
                    break;

            case 5: print(&q);
                    break;

            default: printf("Invalid option");
                     break;
        }
    }while(op!=6);
}

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,int 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,int 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;
        }
}

int dequeueF(dequeue *P)
{
        int 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);
}
```

```
int dequeueR(dequeue *P)
{
	int 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("%d, ",P->data[i]);
		i=(i+1)%MAX;
	}

	printf("%d\n",P->data[P->rear]);
}
```

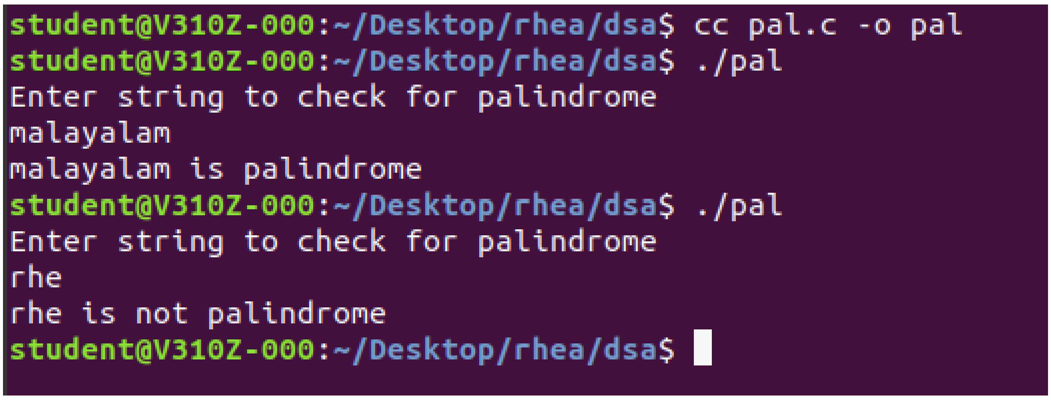


```
student@V310Z-000:~/Desktop/rhea/dsa$ cc deq.c -o deq
student@V310Z-000:~/Desktop/rhea/dsa$ ./deq

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(front)
5.Print
6.Exit

Enter your choice:1

Enter number of elements:4

Enter the data:2 3 5 6

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(front)
5.Print
6.Exit

Enter your choice:2

Enter element to be inserted:7

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(front)
5.Print
6.Exit

Enter your choice:4

Element deleted is 2

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(front)
5.Print
6.Exit

Enter your choice:5
3, 5, 6, 7
```

Q3 PALINDROME STRING USING DEQUEUE

```
#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)

    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 ans=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))
{
        ans = 1;
        break;
}
}
        if(ans == 0)
        printf("%s is palindrome\n",c);
else
        printf("%s is not palindrome\n",c);
        return 0;
}
```



```
student@V310Z-000:~/Desktop/rhea/dsa$ cc pal.c -o pal
student@V310Z-000:~/Desktop/rhea/dsa$ ./pal
Enter string to check for palindrome
malayalam
malayalam is palindrome
student@V310Z-000:~/Desktop/rhea/dsa$ ./pal
Enter string to check for palindrome
rhe
rhe is not palindrome
student@V310Z-000:~/Desktop/rhea/dsa$ 
```