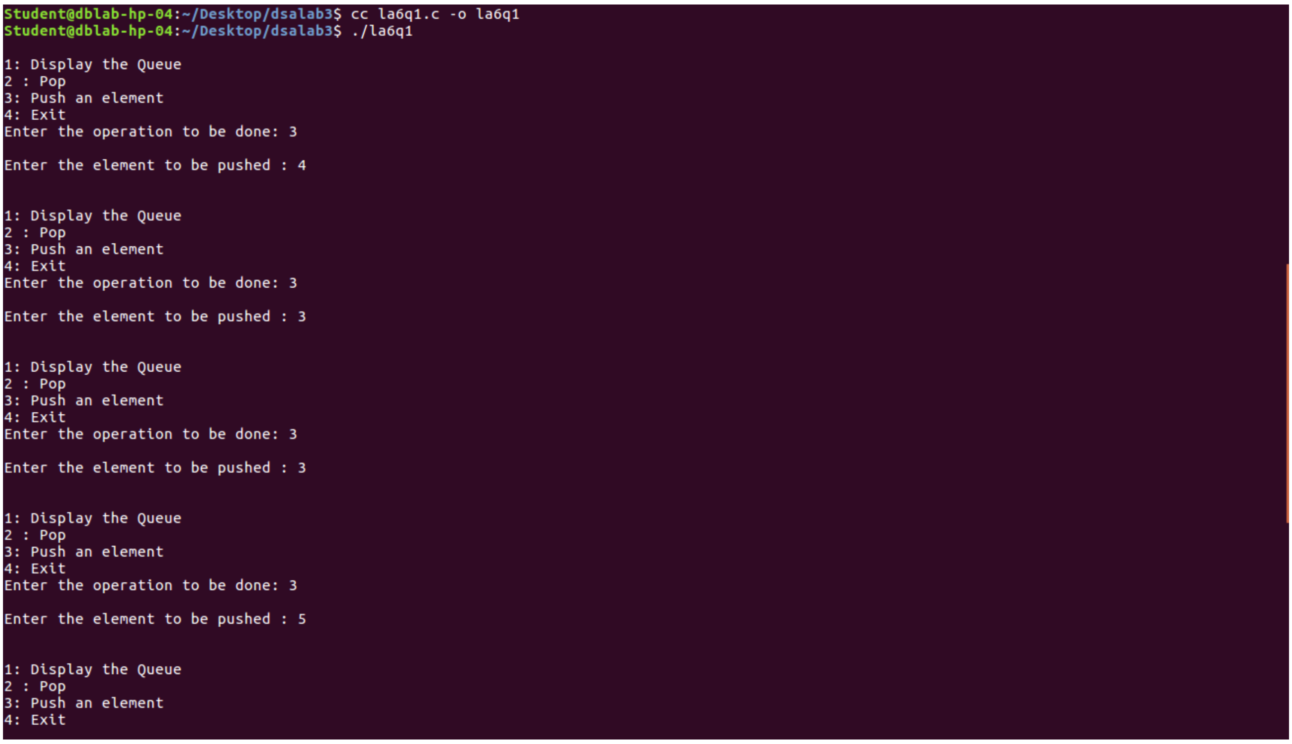
Ayush Goyal
190905522

DSA Lab 6

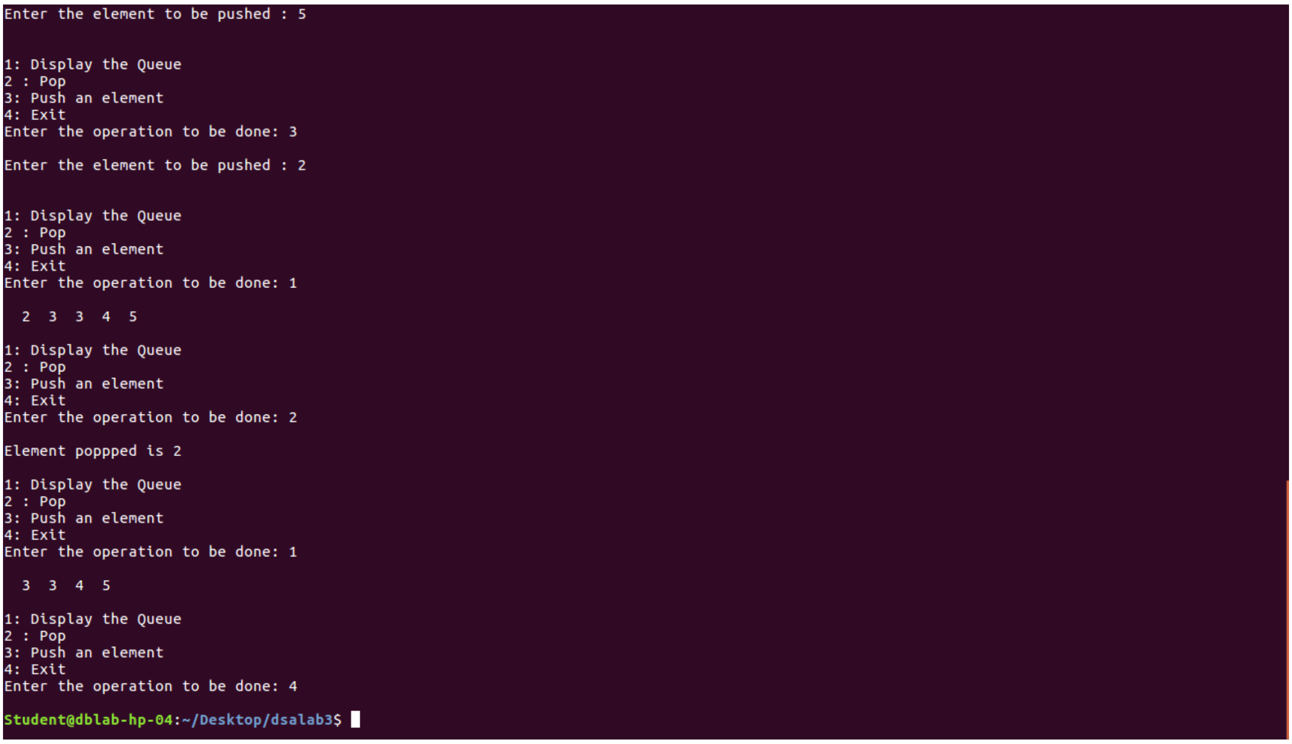
Q1) Implement an ascending priority queue.

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

#define MAX_QUEUE_SIZE 10

typedef struct
{
   int front, rear;
   int array[MAX_QUEUE_SIZE];
} Queue;

void display(Queue q)
{
   if (q.front == -1 && q.rear == -1)
   {
      printf("\nThe queue is empty.");
   }
   else
   {
      printf("\n");
      for (int i = q.front; i <= q.rear; i++)
      {
         printf("%3d", q.array[i]);
      }
   }
}

void push(Queue *q, int key)
{
   if (q->rear == MAX_QUEUE_SIZE - 1)
   {
      printf("\nThe queue is full");
   }
   else
   {
      if (q->front == -1 && q->rear == -1)
      {
         q->front++;
      }
      int pos;
      for(int i = q->front; i<=q->rear; i++){
         if(q->array[i]<=key){
            pos = i+1;
         }
      }
      for(int i = q->rear; i>=pos;i--){
            q->array[i+1] = q->array[i];
        }
        q->rear++;
        q->array[pos] = key;
    }
}



int pop(Queue *q)
{
    int temp = q->array[q->front];
    q->front++;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}

int main()
{
    Queue q;
    q.front = -1;
    q.rear = -1;
    int choice = 0, ele;
    while (choice < 4)
    {
        printf("\n1: Display the Queue \n2 : Pop  \n3: Push an element \n4: Exit");
        printf("\nEnter the operation to be done: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            display(q);
            break;



        case 2:
            if (q.front == -1 && q.rear == -1)
            {
                printf("\nThe queue is empty");
            }
            else
            {
                ele = pop(&q);
                printf("\nElement poppped is %d", ele);
            }
            break;
```

```
         case 3:
            printf("\nEnter the element to be pushed : ");
            scanf("%d", &ele);
            push(&q, ele);
            break;
         }
         printf("\n");
     }
     return 0;
}
```



```
Student@dblab-hp-04:~/Desktop/dsalab3$ cc la6q1.c -o la6q1
Student@dblab-hp-04:~/Desktop/dsalab3$ ./la6q1

1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 4


1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 3


1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 3


1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 5


1: Display the Queue
2 : Pop
3: Push an element
4: Exit
```

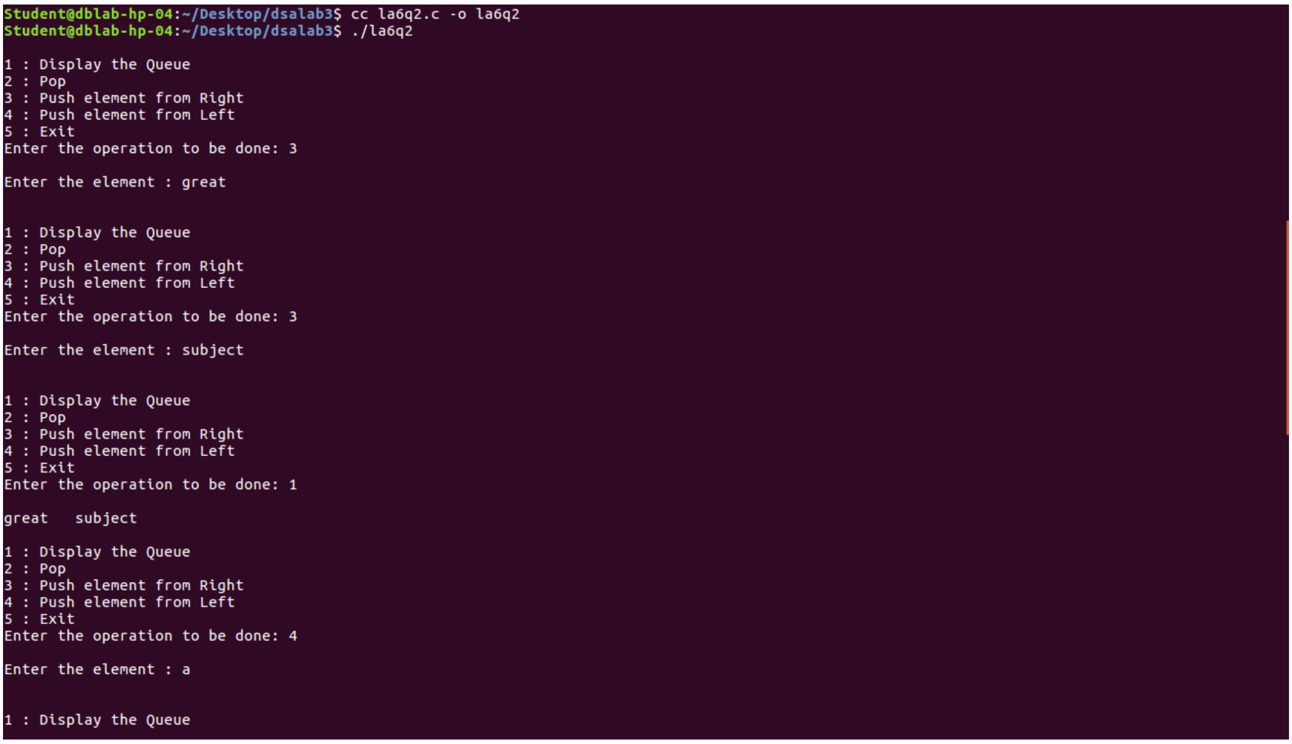


```
Enter the element to be pushed : 5


1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 2


1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 1

  2  3  3  4  5

1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 2

Element poppped is 2

1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 1

  3  3  4  5

1: Display the Queue
2 : Pop
3: Push an element
4: Exit
Enter the operation to be done: 4

Student@dblab-hp-04:~/Desktop/dsalab3$ 
```

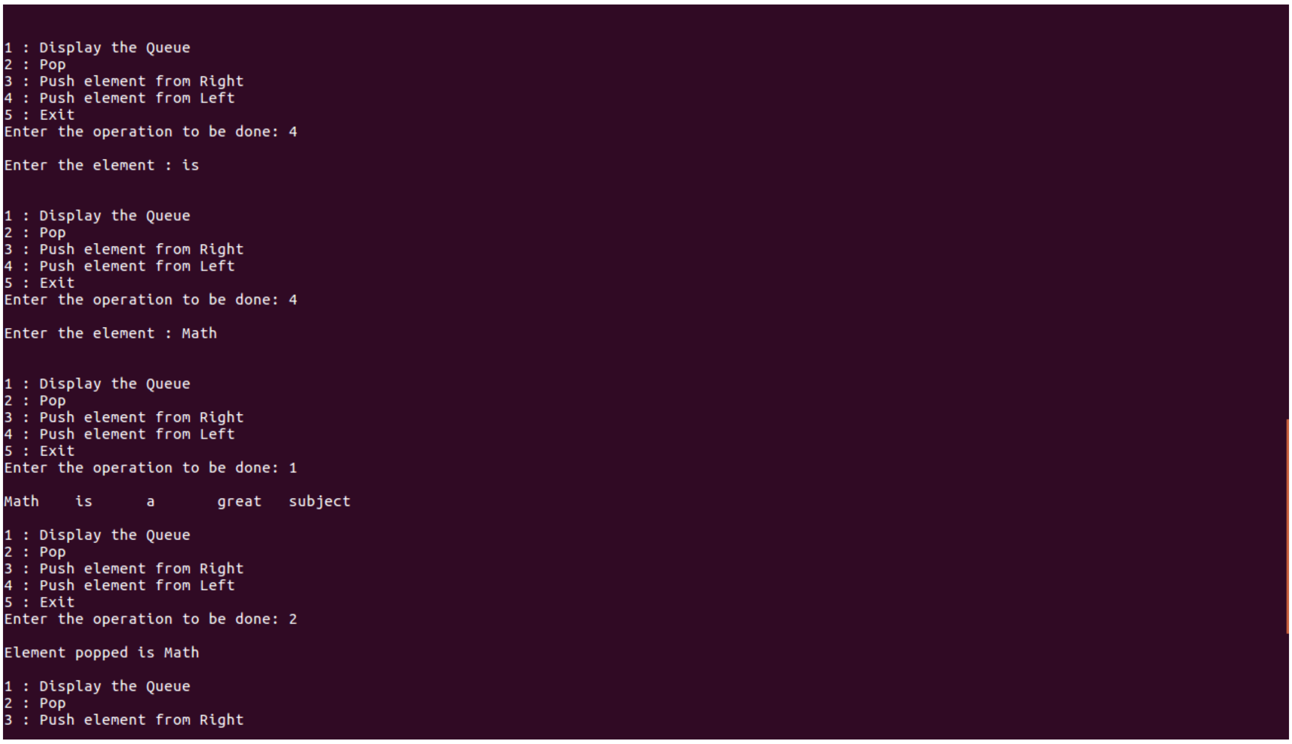
Q2) Implement a queue of strings using an output restricted dequeue (no   deleteRight).

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

#define MAX_QUEUE_SIZE 5

typedef struct
{
   int front, rear;
   char* array[MAX_QUEUE_SIZE];
} Queue;

void print(Queue q)
{
   if (q.front == -1 && q.rear == -1)
   {
      printf("\nThe queue is empty, nothing to print");
   }
   else
   {
      printf("\n");
      for (int i = q.front; i <= q.rear; i++)
      {
         printf("%s\t", q.array[i]);
      }
   }
}

void pushRight(Queue *q, char* key)
{
   if (q->rear == MAX_QUEUE_SIZE - 1)
   {
      printf("\nThe queue is full, cannot push");
   }
   else
   {
      if (q->front == -1 && q->rear == -1)
      {
         q->front++;
      }
      q->array[++q->rear] = key;
   }
}

void pushLeft(Queue *q, char* key)
{
   if (q->rear == MAX_QUEUE_SIZE - 1)
   {
      printf("\nThe queue is full, cannot push");
   }
   else
```

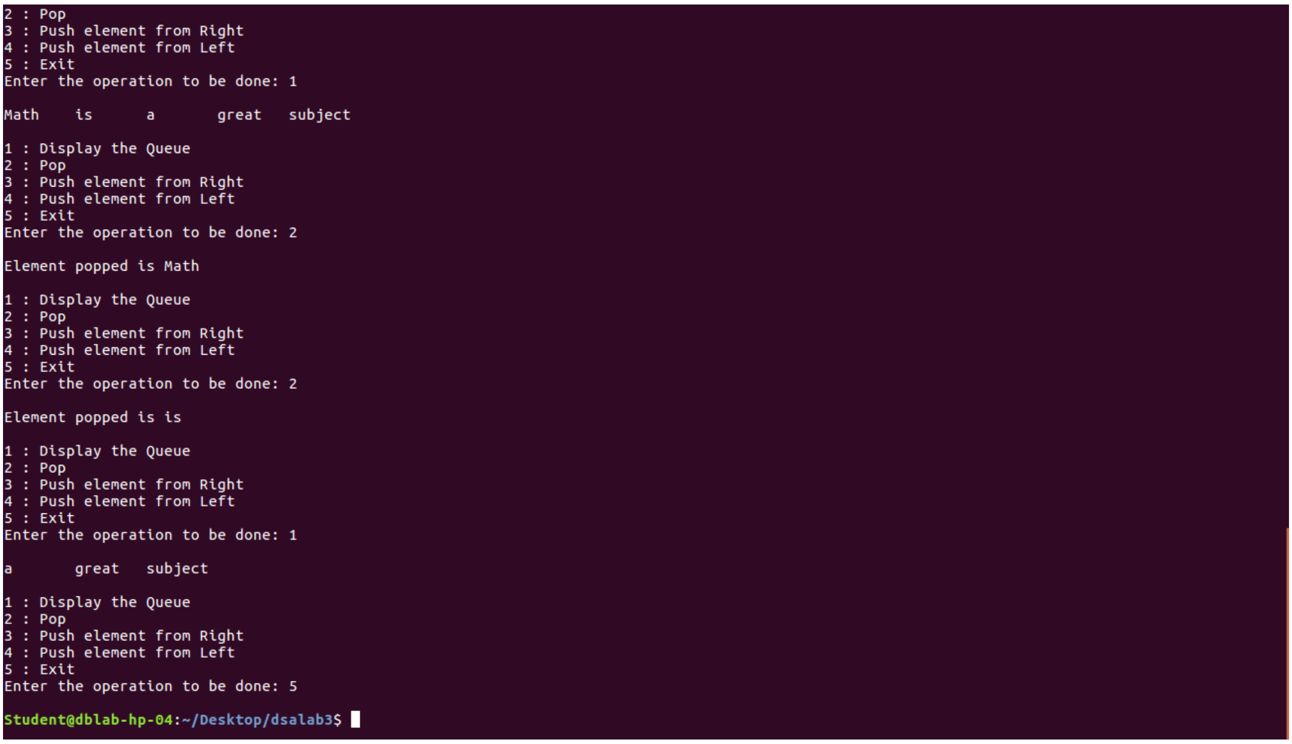
```
    {
        if (q->front == -1 && q->rear == -1)
        {
            q->front++;
        }
        for(int i = q->rear; i>=q->front; i--){
            q->array[i+1] = q->array[i];
        }
        ++q->rear;
        q->array[q->front] = key;
    }
}

char* pop(Queue *q)
{
    char* temp = q->array[q->front];
    q->front++;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}

char* front(Queue q)
{
    return q.array[q.front];
}

int main()
{
    Queue q;
    q.front = -1;
    q.rear = -1;
    int ch = 0;
    char* ele;
    while (ch < 5)
    {
        printf("\n1 : Display the Queue \n2 : Pop \n3 : Push element from Right\n4 : Push element
from Left\n5 : Exit");
        printf("\nEnter the operation to be done: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            print(q);
            break;

        case 2:
            if (q.front == -1 && q.rear == -1)
            {
                printf("\nThe queue is empty");
            }
            else
            {
                ele = pop(&q);
                printf("\nElement popped is %s", ele);
            }
            break;

        case 3:
            ele = (char*)calloc(100, sizeof(char));
            printf("\nEnter the element : ");
            scanf(" %s", ele);
            pushRight(&q, ele);
            break;

        case 4:
            ele = (char*)calloc(100, sizeof(char));
            printf("\nEnter the element : ");
            scanf(" %s", ele);
            pushLeft(&q, ele);
            break;
        }
        printf("\n");
    }
    return 0;
}
```



```
Student@dblab-hp-04:~/Desktop/dsalab3$ cc la6q2.c -o la6q2
Student@dblab-hp-04:~/Desktop/dsalab3$ ./la6q2

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 3

Enter the element : great


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 3

Enter the element : subject


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 1

great   subject

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 4

Enter the element : a


1 : Display the Queue
```



```
1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 4

Enter the element : is


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 4

Enter the element : Math


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 1

Math    is      a       great   subject

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is Math

1 : Display the Queue
2 : Pop
3 : Push element from Right
```

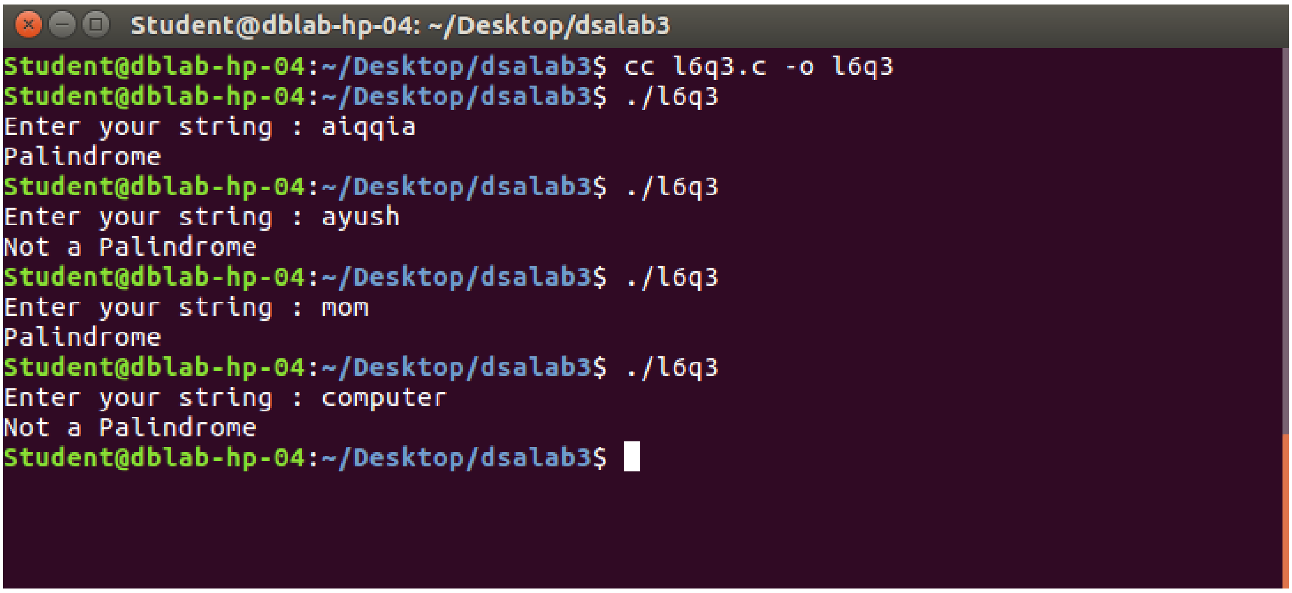


```
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 1

Math    is      a       great   subject

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is Math

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is is

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 1

a       great   subject

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 5

Student@dblab-hp-04:~/Desktop/dsalab3$
```

Q3) 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_QUEUE_SIZE 10

typedef struct
```

```
{
    int front, rear;
    char array[MAX_QUEUE_SIZE];
} Queue;

void pushR(Queue *q, char key)
{
    if (q->rear == MAX_QUEUE_SIZE - 1)
    {
        printf("\nThe queue is full");
    }
    else
    {
        if (q->front == -1 && q->rear == -1)
        {
            q->front++;
        }
        q->array[++q->rear] = key;
    }
}

char popRight(Queue *q)
{
    char temp = q->array[q->rear];
    q->rear--;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}

char popLeft(Queue *q)
{
    char temp = q->array[q->front];
    q->front++;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}

int main()
{
    Queue q;
    q.front = q.rear = -1;
    char ele[100];
    printf("Enter your string : ");
    scanf(" %s", ele);
    int n = strlen(ele);
    for(int i = 0; i<n; i++){
        pushR(&q, ele[i]);
    }
    n = n/2;
    int p = 1;
    while(n--){
        if(popLeft(&q)!=popRight(&q)){
            p = 0;
            break;
        }
    }
    if(p){
        printf("Palindrome\n");
    }
    else{
        printf("Not a Palindrome\n");
    }
    return 0;
}
```



```
Student@dblab-hp-04:~/Desktop/dsalab3$ cc l6q3.c -o l6q3
Student@dblab-hp-04:~/Desktop/dsalab3$ ./l6q3
Enter your string : aiqqia
Palindrome
Student@dblab-hp-04:~/Desktop/dsalab3$ ./l6q3
Enter your string : ayush
Not a Palindrome
Student@dblab-hp-04:~/Desktop/dsalab3$ ./l6q3
Enter your string : mom
Palindrome
Student@dblab-hp-04:~/Desktop/dsalab3$ ./l6q3
Enter your string : computer
Not a Palindrome
Student@dblab-hp-04:~/Desktop/dsalab3$
```