Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of AI-ML

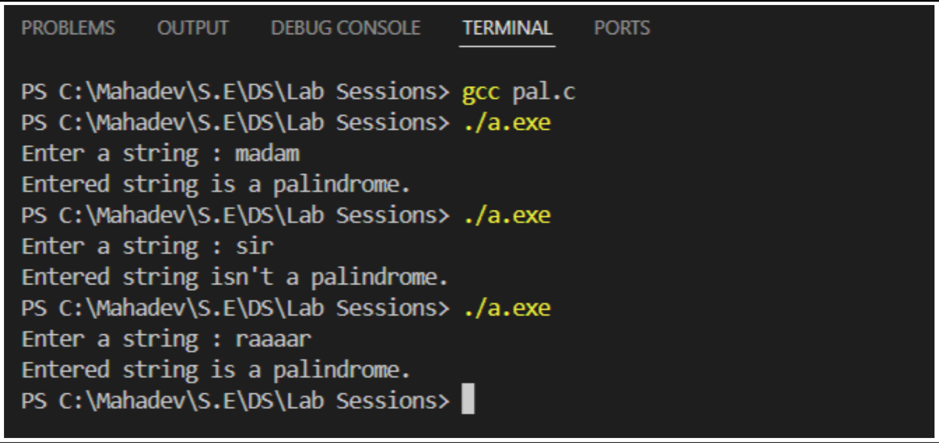
Experiment	2
Aim	Implement the given problem statement
Objective	Palindrome Detection Given an input string, check whether it is a palindrome (reads the same whether read from left to right or from right to left) using queues. You are allowed to use one or multiple queues but do not use a stack. Sample input 1: madam Output: Yes Sample input 2: sir Output: No
Name	Balla Mahadev Shrikrishna
UCID	2023300010
Class	A
Batch	A
Date of Submission	23-08-24

Explanation of the technique used	Created two queues - q1: Used to store the characters of the input string in reverse order. q2: Used to store the characters of the input string in their original order. The characters are then dequeued from both queues one by one, and each pair of characters is compared. If all the characters match, the input string is a palindrome; otherwise, it isn't a palindrome.
Program(Code)	see below

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

#define MAX 50

typedef struct{
    char* arr;
    int front, rear;
}Queue;

Queue* createQueue(){
    Queue* q = (Queue*)malloc(sizeof(Queue));
    q->arr = (char *)malloc(MAX * sizeof(char));
    q->front = 0, q->rear = -1;
    return q;
}
```

```
bool isFull(Queue *q){
    return q->rear == MAX-1;
}

bool isEmpty(Queue *q){
    return q->rear < q->front;
}

void enqueue(Queue *q, char c){
    if(isFull(q)){
        printf("Queue is full...can't enqueue.\n");
        return;
    }
    else{
        q->rear = q->rear + 1;
        q->arr[q->rear] = c;
    }
}

char dequeue(Queue *q){
    if(isEmpty(q)){
        printf("Queue is empty...can't dequeue.\n");
        return '\0';
    }
    else{
        char t = q->arr[q->front];
        q->front = q->front + 1;
        return t;
    }
}

void freeQueue(Queue *q){
    free(q->arr);
    free(q);
}

bool isPalindrome(Queue* q1, Queue* q2, char str[]){
    int len = strlen(str);

    for(int i=0; i<len; i++){
        enqueue(q1,str[len-i-1]);
        enqueue(q2,str[i]);
    }

    while(!isEmpty(q1) && !isEmpty(q2)){
        if(dequeue(q1)!=dequeue(q2)){
            return false;
        }
    }
    return true;
}

int main(){
    char str[MAX];
    Queue *q1 = createQueue();
    Queue *q2 = createQueue();
    printf("Enter a string : ");
    scanf("%s", str);
    if(isPalindrome(q1, q2, str)){
        printf("Entered string is a palindrome.\n");
    }
    else{
        printf("Entered string isn't a palindrome.\n");
    }
    freeQueue(q1);
    freeQueue(q2);
    return 0;
}
```

Output



```
PS C:\Mahadev\S.E\DS\Lab Sessions> gcc pal.c
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter a string : madam
Entered string is a palindrome.
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter a string : sir
Entered string isn't a palindrome.
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter a string : raaaar
Entered string is a palindrome.
PS C:\Mahadev\S.E\DS\Lab Sessions> 
```

Conclusion

Learned how to implement a palindrome detection algorithm using queues. It highlighted the flexibility of queue data structure in solving problems typically associated with stacks.