Data Structure Lab

Assignment No-6/7

Name : Sanket Shivaji Jadhav.

Prn: 2020BTECS00005

1. Developing ADT for Stack and Queue and their applications

```
// 1.Stack and its Applications.
#include<stdio.h>
#include<stdlib.h>
struct stack
{
    int *arr;
    int top;
    int size;
};

// check the initiation status of stack
void isEmpty(struct stack *s){
    if(s->top==-1){
    printf("Stack is a Empty\n");
    }
    else{
        if(s->top==(s->size-1)){
            printf("Stack is Full\n");
        }
        else{
            printf("Stack is Initiated\n");
        }
```

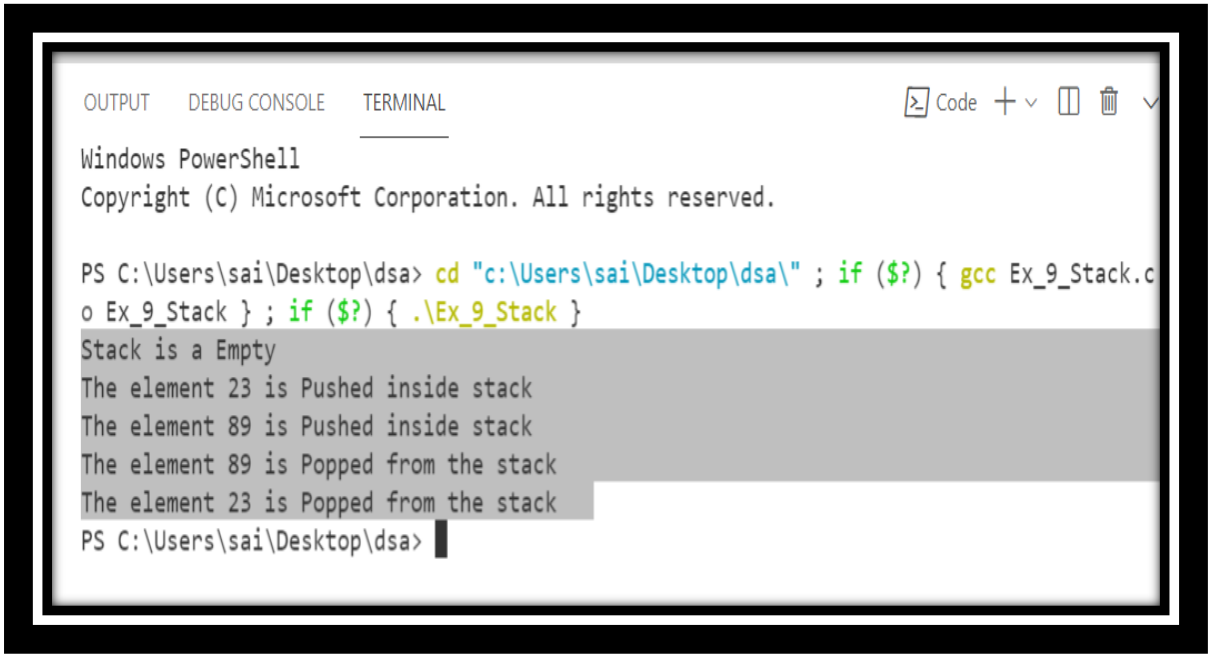
```
    }
}
// to push inside the stack
void push(int d,struct stack *s){
    if(s->top==(s->size-1)){
        printf("Stack Overflow\n");
    }
    else{
        s->top++;
        s->arr[s->top]=d;
        printf("The element %d is Pushed inside
stack\n",(s->arr[s->top]));
    }
}
// to pop in a stack
void pop(struct stack *s){
    if(s->top==-1){
        printf("Stack is Underflow\n");
    }
    else {
        printf("The element %d is Popped from the
stack\n",(s->arr[s->top]));
        s->top--;
    }
}

// main function
int main(){
    struct stack *s=(struct stack*)malloc(sizeof(struct
stack));
    s->size=8;
    s->top=-1;
    s->arr=(int *)malloc(s->size*sizeof(int));
    // to check the stack is empty or full
```

```
    isEmpty(s);
    // push an element in an stack
    push(23,s);
    push(89,s);
    // pop an element inside a stack
    pop(s);
    pop(s);

    return 0;
}
```

OUTPUT :




```
// 2.Queue and its applications
#include <stdio.h>
#include <stdlib.h>
struct queue{
int rare,top,size;
int *arr;
```

```
};

int isempty(struct queue *a){
    if(a->top==-1){
        return 1;
    }
   return 0;
}

int isfull(struct queue *a){
    if(a->rare==(a->size-1)){
        return 1;
    }
    return 0;
}

  // To push the data into the queue.
void enqueue(int n,struct queue *s){
    if(!isfull(s)){
        if(s->top==-1 && s->rare==-1){
           s->top=0;
           s->rare=0;
           s->arr[s->rare]=n;
  printf("The element %d is Enqueued\n",s->arr[s->rare]);
 }
        else{
            s->rare++;
            s->arr[s->rare]=n;
            printf("The element %d is Enqueued\n",s-
>arr[s->rare]);
        }
    }
    else{
        printf("Queue is Full\n");
```

```
    }
}
  // To pop the data into the queue.
void dequeue(struct queue *s){
    if(s->top!=s->size){

        int n=s->arr[s->top];
        s->top++;
        printf("The element %d is dequeued\n",n);

    }
    else{
        printf("Queue is empty\n");
    }
}
// To display the data from queue.
void display(struct queue *s,int size){
    if(isempty(s)){
        printf("Queue is empty\n");
    }
    else{
    for(int i=s->top;i<=s->size-1;i++){
        printf("%d ",s->arr[i]);
    }
    printf("\n");
    }
}

int main(){
    struct queue *a=(struct queue*)malloc(sizeof(struct
queue));
    a->arr=(int *)malloc(sizeof(int));
    a->top=-1;
    a->rare=-1;
```

```
    a->size=5;
    // To push the data into the queue.
    enqueue(23,a);
    enqueue(3,a);
    enqueue(12,a);
    enqueue(87,a);
    enqueue(5,a);
    display(a,a->size);
    // T0 pop the element in an queue
    dequeue(a);
    dequeue(a);
    dequeue(a);
    dequeue(a);
    // To Display elements of an queue
    display(a,a->size);
    return 0;
}
```

OUTPUT :