**Experiment No.1**

* **Problem Statement**: Implement a program on stack ADT and write function for reversing a string
* Theory:

A stack in C is a linear data structure that follows the Last-In-First-Out (LIFO) principle. It means that the last element added to the stack will be the first one to be removed. Think of a stack as a collection of items, similar to a stack of books where you can only add or remove books from the top of the stack. In C, a stack can be implemented using arrays or linked lists.

Here are some key concepts and operations related to a stack in C:

**Push**: Adding an element onto the stack.

**Pop**: Removing an element from the top of the stack.

**Peek**: Viewing the top element without removing it.

**isEmpty**: Checking if the stack is empty.

**isFull**: Checking if the stack is full (for fixed-size stacks).Display: Printing the contents of the stack.

**To reverse a string using stack**

Iterate through the characters in the input string from left to right.For each character, push it onto the stack.After processing all characters, pop each character from the stack and append it to the result string.The result string now contains the reversed string.

* **Algorithm:**

**To reverse a string using stack**

1. Start

2. Initialize an integer 'top' to -1 and an integer array 'stack' with a maximum size of 'max'.

3. Define a 'push' function that takes a character 'x' as input.

a. Check if 'top' is equal to 'max - 1'.

i. If true, print "Stack overflow" and exit the function.

ii. If false, increment 'top' by 1 and set 'stack[top]' to 'x'.

4. Define a 'pop' function.

a. Check if 'top' is equal to -1.

i. If true, print "Stack underflow" and exit the function.

ii. If false, print the character at 'stack[top]' and decrement 'top' by 1.

5. In the 'main' function:

a. Declare a character array 'str' with a maximum size of 'max'.

b. Prompt the user to enter a string and store it in 'str' using 'scanf'.

c. Calculate the length of the string using 'strlen' and store it in 'len'.

d. Declare an integer 'i'.

6. Iterate from 'i' equals 0 to 'i' less than 'len':

a. Call the 'push' function with 'str[i]' as the argument to push each character onto the stack.

7. Print "Reversed string: ".

8. Iterate from 'i' equals 0 to 'i' less than 'len':

a. Call the 'pop' function to pop characters from the stack and print them.

9. End.

* **Program:**

***#include* <stdio.h>**

***#include* <stdlib.h>**

**struct Stack**

**{**

**int top;**

**int size;**

**char \*array;**

**};**

**struct Stack \*initialize\_stack(int size)**

**{**

**struct Stack \*stack = (struct Stack \*)malloc(sizeof(struct Stack));**

**stack->size = size;**

**stack->top = -1;**

**stack->array = (char \*)malloc(size \* sizeof(char));**

***return* stack;**

**}**

**int isFull\_stack(struct Stack \*stack)**

**{**

***return* stack->top == stack->size - 1;**

**}**

**int isEmpty\_stack(struct Stack \*stack)**

**{**

***return* stack->top == -1;**

**}**

**void push(struct Stack \*stack, char item)**

**{**

***if* (isFull\_stack(stack))**

**{**

**printf("Stack Overflow\n");**

***return*;**

**}**

**stack->array[++stack->top] = item;**

**printf("Pushed element: %c\n", item);**

**}**

**char pop(struct Stack \*stack)**

**{**

***if* (isEmpty\_stack(stack))**

**{**

**printf("Stack Underflow\n");**

***return* 0;**

**}**

***return* stack->array[stack->top--];**

**}**

**void display(struct Stack \*stack)**

**{**

***if* (isEmpty\_stack(stack))**

**{**

**printf("Stack is empty\n");**

**}**

***else***

**{**

**printf("Stack elements: \n");**

***for* (int i = stack->top; i >= 0; i--)**

**{**

**printf("%c \n", stack->array[i]);**

**}**

**printf("\n");**

**}**

**}**

**void peek(struct Stack \*stack)**

**{**

***if* (isEmpty\_stack(stack))**

**{**

**printf("Stack is empty\n");**

**}**

***else***

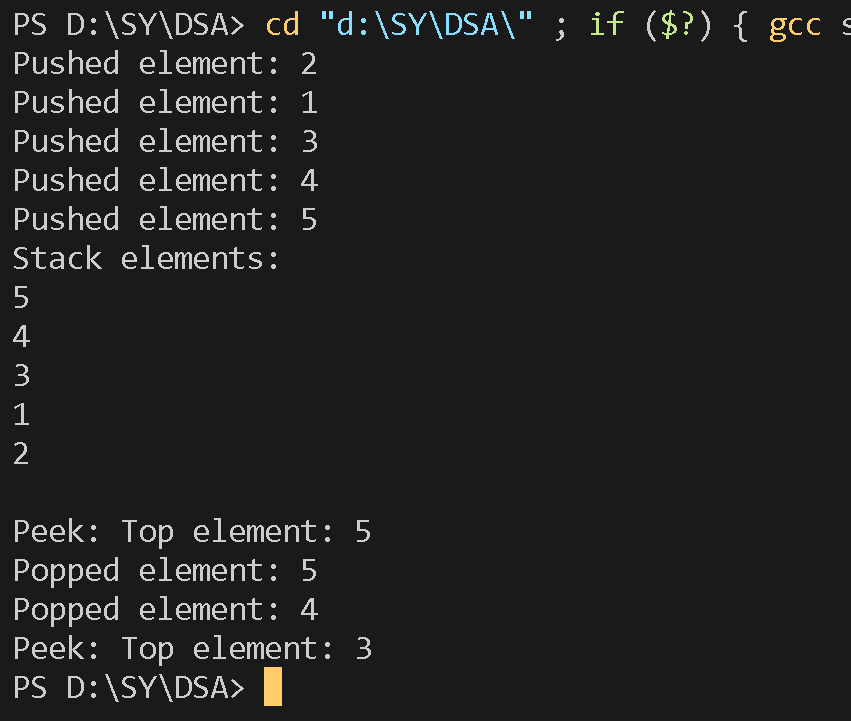
**{**

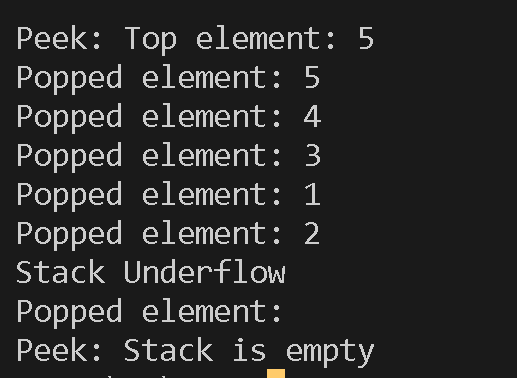
**printf("Top element: %c\n", stack->array[stack->top]);**

**}**

**}**

**Output:**





***/\****

***\* File: reverseString.c***

***\* Author: Manjiri Chavande***

***\* Email: cmanjiri1912@gmail.com***

***\* Created: September 6, 2023***

***\* Description: Reversing a string***

***\*/***

***// reverse\_a\_string.c***

***#include* <stdio.h>**

***#include* <stdlib.h>**

***#include* <string.h>**

***#include* "stacks.c"**

***// Function to reverse a string using the stack***

**void reverse\_string(char \*msg){**

**struct Stack \*stack = initialize\_stack(strlen(msg));**

***// Push each character of the string onto the stack***

***for* (int i = 0; i < strlen(msg); i++){**

**push(stack, msg[i]);**

**}**

**printf("Reversed String: ");**

***// Pop and print each character to reverse the string***

***while* (!isEmpty\_stack(stack)){**

**printf("%c", pop(stack));**

**}**

**printf("\n");**

**}**

**int main(){**

**reverse\_string("SPIT Comps");**

***return* 0;**

**}**



* **Conclusion**: Thus we have successfully implemented stacks and performed operations on it , and also revered a string using stacks.