**FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY)**

****

**COMP 451 (Compiler Construction)**

**2022 FALL**

**Lab - 9**

**Gulraiz Noor Bari (231-525536)**

**INTRODUCTION:**

* **stdio.h** (standard input/output)is a header file that contains declarations for functions like, printf, scanf, etc.
* **stdlib.h** (standard library) is a header file that contains declarations of functions that involves memory allocation and process control. For example, in our program we used **exit(0)** at the end; which means to successfully terminate the program.
* **string.h** is a header file that contains declarations of functions that are used for working with strings. In our program we have used **strlen** function to get the length of string.
* **ctype.h** is a header file that contains functions that are used to handle characters.
* **main():** The main function serves as the starting point of the program execution in C language. User can pass any number of parameters depending upon the requirements of the program logic or structure.

**LOGIC/ALGORITHM:**

The code is a program that implements Operator Precedence Parser for the following grammar:

**E 🡪 E + E**

**E 🡪 E \* E**

**E 🡪 id**

* The program starts by checking whether two arguments were passed by the user in command line, which is done by argc (which has the count of arguments entered in the command-line). If the number of arguments is not two the function is not proceeded and displays a message “Invalid argument”, otherwise if user has entered two arguments; in our case the executable and the string to be parsed, the program proceeds.
* We initialize three lists that will be implemented to behave as stacks, throughout the program and initialize three pointers and each of them will point to the top of each stack.
* Then we make a function in which takes a character as input and returns its corresponding precedence, which is done using switch statements. The precedence of the operators is:

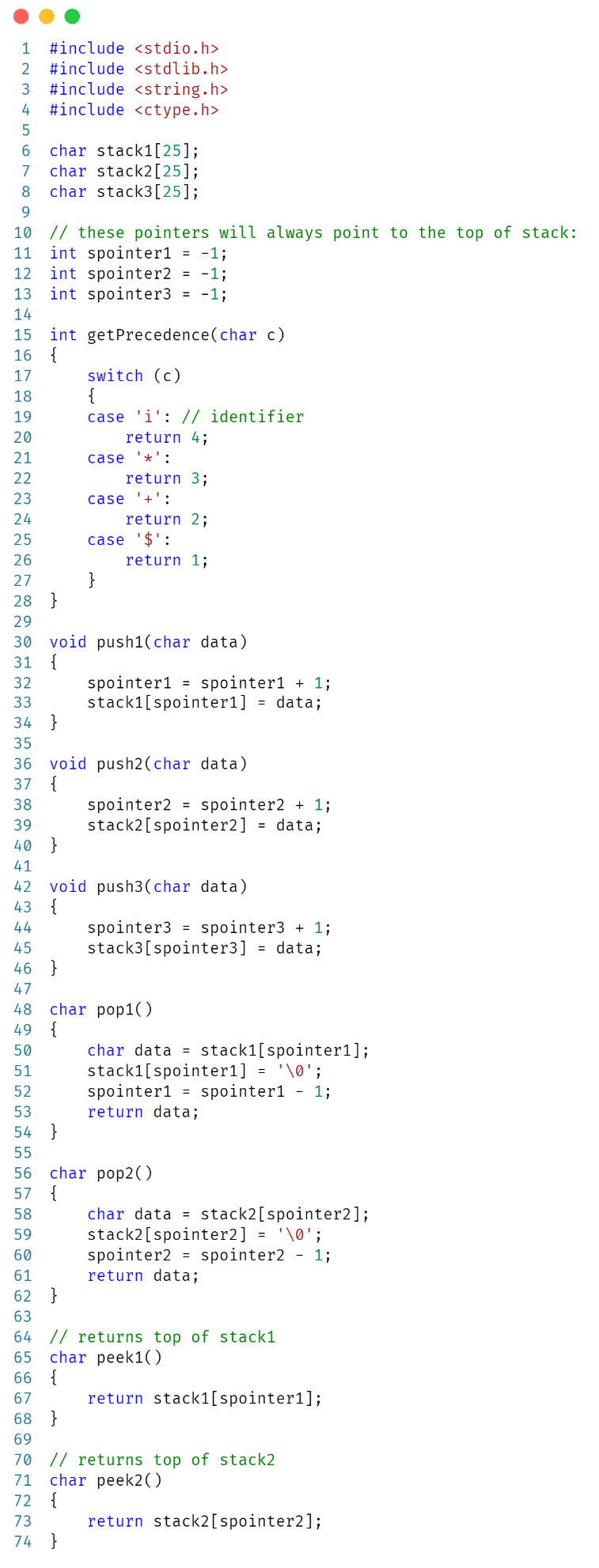
**Identifier 🡪 4 (highest)**

**\* 🡪 3**

**+ 🡪 2**

**$ 🡪 1 (lowest)**

* Then we write functions for the implementation of stack features:
  + Push: pushing elements on the top of stack
  + Pop: removing elements from the top of stack
  + Peak: check the element on the top of stack
* The main function is initialized and **“$”** is pushed on stack1 and the input string entered by user is push in reverse order (so that the first character of the string is on the top of stack) on the stack2.
* In the infinite while loop we implement the main logic of the operator precedence parser, which is as follows:
  + If top of stack1 > top of stack2, then pop from stack1 and push it on stack3 (output stack)
  + If top of stack1 < top of stack2, then pop from stack2 and push it on stack1
* The loop and the program end when both the stacks (stack1 and stack2) have the top pointers pointing to **“$”** (terminal symbol); as it will always be at the bottom of the stack).





**OUTPUT:**

