

Problem Definition

You are asked to accept an infix string, representing a simple arithmetic expression, from the user, transform the string to prefix form and evaluate prefix expression.

Input: The input is a string containing only **single digit numbers**, arithmetic operators represented by `+` `-` `*` `/` by their conventional meanings. Parentheses **will not be used** in the input. You can assume that the input contains no semantical or syntactical errors.

Given: Three helper functions are provided. `higher_priority()` function calculates and returns the operator with higher priority or not. `evaluate()` function calculates the result of applying its third parameter as an arithmetic operation over its first and second parameters. `str_reverse()` function takes a string parameter and reverses this string expression.

Wanted: You will complete three functions in the code as outlined below. Infix expression is taken from users in `main()` function. After that, `'infix_to_prefix()'` function is called. You should do following transformation in these functions.

- `void infix_to_prefix(string infix, string& prefix):`
 - Reverse infix string (you can use `str_reverse()` function in the code)
 - Called `infix_to_postfix()` function and give this reversed infix string and prefix string as parameters
 - Reverse prefix string
- `void infix_to_postfix(string infix, string& postfix):`
 - Scan the infix expression from left to right
 - If the scanned character is the number then this number is concatenated to postfix string
 - If the scanned character is an operator and the stack is not empty:
 - * If the priority of the top most operator of the stack is less than or equal to scanned operator, push this operator into the stack
 - * If the priority of the top most operator of the stack is greater than scanned operator, pop the operator from the stack and concatenate this operator to postfix string until find a low or equal priority operator
 - If the scanned character is an operator and the stack is empty then push this operator into the stack
 - When the infix string ends pop all the operators remaining in the stack and concatenate to postfix string
- `void eval_prefix(string expr):`
 - Scan the prefix expression from **right to left**
 - Push number into stack
 - For an operator pop two numbers from stack, apply the operator and push the number back to stack
 - When the prefix string ends, pop the single number in the stack as the result

1 Example

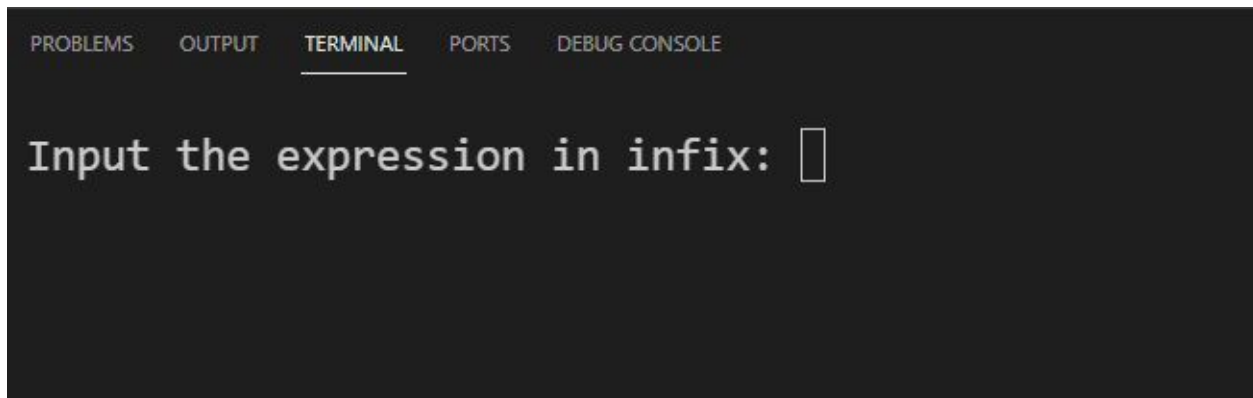


Figure 1: Terminal screen when the program is run.

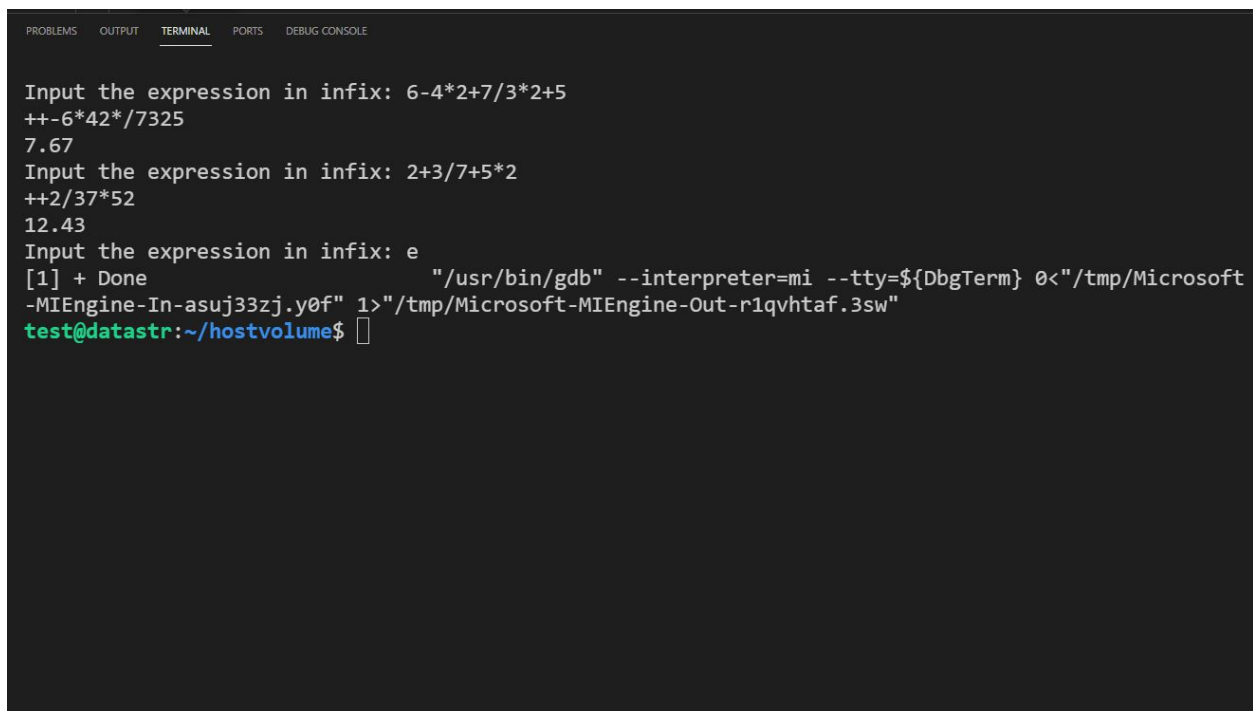


Figure 2: Terminal screen when the input is entered.

Submission Rules

- Make sure you write your name and number in all of the files of your project, in the following format:
/* @Author
Student Name: <student_name>
Student ID : <student_id>
Date: <date> */
- Use comments wherever necessary in your code to explain what you did.
- Your program will be checked by using **Calico**(<https://bitbucket.org/uyar/calico>) automatic checker.

- Do not share any code or text that can be submitted as a part of an assignment (discussing ideas is okay).
- Only electronic submissions through Ninova will be accepted no later than deadline.
- You may discuss the problems at an abstract level with your classmates, but you should not **share or copy code** from your classmates or from the Internet. You should submit your **own, individual** homework.
- Academic dishonesty, including cheating, plagiarism, and direct copying, is unacceptable.
- If you have any question about the recitation, you can send e-mail to Yunus Emre Cebeci(cebeci16@itu.edu.tr).
- Note that **YOUR CODES WILL BE CHECKED WITH THE PLAGIARISM TOOLS!**



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.