Atılım University
Department of Computer Engineering

CMPE 226 Data Structures
Fall, 2019-2020
Homework 2
Due Date: December 20th, 2019

1. In computer science, an arithmetic expression can be written as either a postfix, infix or prefix expression. You can see the examples for each of these types. Notice that the individual numbers and operators are separated with one space.

   Infix: 12 + **(**2 * 5**)** + 3 => 25 → 12 + 2 * 5 + 3 => 25

   Postfix: **{[**12 **(**2 5 ***)** +**]** 3 +**}** => 25 → 12 2 5 * + 3 + => 25

   Prefix: **{**+ **[**+ 12 **(*** 2 5**)]** 3**}** => 25 → + + 12 * 2 5 3 => 25

   An interesting observation is the order of operands are same for all types of expressions. However, same observation does not fit to the order of operators. Knowing that, you can convert each expression to another manually. You can see a step by step example which converts an arithmetic expression from infix form to prefix form.

   Step 1: 32 + 5 * 12 – 3 * 13

   Step 2: 32 + (5 * 12) – (3 * 13)

   Step 3: 32 + (* 5 12) - (* 3 13)

   Step 4: (32 + (* 5 12)) - (* 3 13)

   Step 5: (+ 32 (* 5 12)) - (* 3 13)

   Step 6: - (+ 32 (* 5 12)) (* 3 13)

   Step 7: - + 32 * 5 12 * 3 13

   Your first job for this question is to write a function that implements a stack based algorithm which performs a conversion from infix to prefix form on a given string as a parameter. You may want to make a little research about the topic on the Internet. The prototype of the function should look like:

   ```
   char* infix_to_prefix(const char* infix_str);
   ```

   Your second job is to **evaluate the returned prefix expression** from the `infix_to_prefix` function and write the result to the terminal. Your program must take the **input (which are the lines of the text file) from the provided file to a queue**. Each line in the file contains an expression. Then, you can take each element from the queue and convert to prefix form and evaluate the result.

```
Infix Form: 65 + 23 – 47 * 56 * 2 / 5
Prefix Form: - + 65 23 / * 47 * 56 2 5
Result: -964
============================
Infix Form: 32 + 5 * 12 – 3 * 13
Prefix Form: - + 32 * 5 12 * 3 13
Result: 53
============================
… 48 more ...
============================
```

You must follow the operator precedence rules to synchronize your stack contents of the C programming language. Your output should look **exactly** like to the above sample run.

Your program should run similar for different text files that have the same format. Thus, write your programs as generic as possible. Your program will be tested with a different 50 arithmetic expressions.

Cheating and group work is strictly forbidden!

Late submissions will not be tolerated!