Assignment Questions: Stacks and Queues

Information:

Program templates for questions 1-4 are given as separated files (**Q1_template_SQ.c, Q2_template_SQ.c**, **Q3_template_SQ.c**, and **Q4_template_SQ.c**). You **must use** them to implement your functions. The program contains a *main()* function, which includes a switch statement to execute different functions that you should implement. Each function can be called multiple times depending on the user's choice. You need to submit your code to the **https://www.hackerearth.com** (Email invitation will be sent to your school account). You need to submit your code to the NTULearn (**State your name and your lab group in your submission**). Deadline for program submission: **February 25th, 2024 (Sunday) 11.59 pm**.

.

Assignment Questions (1-4)

1. Write a c function interleaveQueue()that rearranges the elements by interleaving the first half of the queue with the second half of the queue using **one temporary stack** for a given **queue of integers of even length**. Note that the interleaveQueue() function only uses **push()** and **pop()** when adding or removing integers from the stack and only uses **enqueue()** and **dequeue()** when adding or removing integers from the queue. [**Please note that you must use only one temporary stack inside your function.**]

   The function prototypes are given as follows:

   ```
   void interleaveQueue(Queue *q)
   ```

   Some test sample cases:
   If the queue is (**1, 2, 3, 4, 5, 6**), the resulting queue will be (**1, 4, 2, 5, 3, 6**)
   if the queue is (**1, 2, 3, 4**), the resulting queue will be (**1, 3, 2, 4**)

2. Write a C program to convert an **infix expression to a prefix expression**. The input and the output of the function are character strings. The input expression contains only four possible operators: **+, -, *** and /. Operands can be any alphanumeric. Each operand is represented by a character symbol. The parentheses are allowed in the input expression. You may assume that the expression is always valid.

   The function prototypes are given as follows:

   ```
   void infixtoPrefix(char* infix, char* prefix);
   ```

   Some test sample cases:

   ```
   Infix: (A+B)+(C-D)
   Prefix: ++AB-CD

   Infix: a+b*c-d*(e/f)
   Prefix: -+a*bc*d/ef

   Infix: 9+(8-7)*(6/(5-4)+3)
   Prefix: +9*-87+/6-543
   ```

3. Write a C program to convert an infix expression to a postfix expression. The input and the output of the function are character strings. The input expression contains only four possible operators: +, -, * and /. Operands can be any alphanumeric. Each operand is represented by a character symbol. The parentheses are allowed in the input expression. You may assume that the expression is always valid.

The function prototype is given as follows:

```
void infixtoPostfix(char* infix, char* postfix);
```

Some test sample cases:

```
Infix: (A+B)+(C-D)
Postfix: AB+CD-+

Infix: a+b*c-d*(e/f)
Postfix: abc*+def/*-

Infix: 9+(8-7)*(6/(5-4)+3)
Postfix: 987-654-/3+*+
```

4. Write a C program to evaluate a postfix expression. You may assume that **operands are single digit numbers.**

The function prototype is given as follows:

```
double postfixEvaluation(char* postfix);
```

Some test sample cases:

```
Input: 987-654-/3+*+
Output: 18.00

Input: 25+89/9*7*+
Output: 63.00

Input: 88/8*77+7+5*6*-
Output: -622.00
```