Homework 2 – Stacks and Queues
TA: WenJ (asdfg123560yahoo@gmail.com)
Deadline: 2019 Oct.29, 11:59pm

1. Objective
   - ☐ Determine if a given string is a valid expression using a stack. If an expression is valid, you need to show its postfix form and evaluate its answer using a stack as well.
   - ☐ Implement a queue function that determines the minimum time required for the maximum number of the oranges becoming rotten.

2. Descriptions
   (1) Validation and evaluation of expressions
   - ✧ Function details:
     Your program should read the standard input file that stores a sequence of strings containing the operators of "(", ")", "+", "-", "*", "/", and non-negative digits of "1", "2", "3", "4", "5", "6", "7", "8", "9", "0". Your program needs to check whether each expression in the file is valid using a stack. For example, the following are some invalid expressions.

1++2
(1*1))
((1-1)
1+2-
-1+1
12/1
)5*4
3*2(

     If one expression is valid, print out its corresponding postfix form and evaluate the answer by using a stack.

   - ✧ Input format:
     - A file includes expressions which may include the operators of "(", ")", "+", "-", "*", "/", and 0-9 digits.
     - The first line of the input file is the number of expressions.
     - Each of the second line to the last line in the input file is one expression.
     - The following is a sample of input format for the input file.

Input
5 (1+2)-(4+1*8) (5+5)-8/4 ((8+9) 8+9-(6/3) 1*2/3*(5+6))

   - ✧ Output format:
     - The output consists of the validation of an expression with its corresponding postfix expression. If the expression is valid, you need to print "1"; otherwise, print "0". Furthermore, if your validation is valid, you need to show its right postfix form and evaluate the answer of the expression. You need to separate the validation status, the postfix form, and the expression answer with a space.

- The following is a sample of the output format. (The red color is the answer to the corresponding expression):

Output
1 12+418*+- -9 1 55+84/- 8 0 1 89+63/- 15 0

(2) Minimum time required to rot the maximum number of the oranges

✧ Function details:

Your program should read a standard input file that stores a matrix of dimension m*n, where each cell in the matrix can have values 0, 1, or 2 which has the following meanings:

0: an empty cell 1: a cell with fresh oranges 2: a cell with rotten oranges

You have to determine the minimum time required for the maximum oranges becoming rotten. A rotten orange at index [i, j] can rot other fresh oranges at indexes [i-1, j], [i+1, j], [i, j-1], [I, j+1] (up, down, left, and right).

Following is the required method that uses a queue to do this task:

1) Create an empty queue, and initialize the count of the time frame to 0.
2) Find all rotten oranges and insert them to a queue (from left to right, from top to down in a matrix). Furthermore, insert a delimiter to indicate the beginning of the next time frame.
3) While Q is not empty do following
   a) Do following while delimiter in the queue is not reached
      ◇ Dequeue an orange from the queue, and rot all adjacent (up, down, left, and right) oranges and insert them into the queue. (Note that the rotten oranges that have been enqueued earlier cannot be enqueued again.) While rotting the adjacent, make sure that the count of the time frame is incremented only once.
   b) Dequeue the old delimiter and enqueue a new delimiter as the queue is not empty. The oranges rotten in the previous time frame lie between the two delimiters.

✧ Input format:

- An input file stores a matrix of dimension m*n where each cell in the matrix can have values 0, 1, 2.
- The first line in the input file is the value of matrix row.
- The second line in the input file is the value of matrix column.
- The third line to the last line of the input file is the m*n matrix.
- The following is a sample of input format for the input file.

Input
3 5 2 1 0 2 1 1 0 1 2 1 1 0 0 2 1

✧ Output format:

- The output needs to print all the steps of enqueue and dequeue operations as follows

```
en(x, y)
de(x, y)
```

, and you need to follow this direction sequence: up->down->left->right

- You also need to show the number of minimum time required to rot the maximum number of the oranges.
- Last, you need to print 1, if all the oranges have become rotten.
  or print 0 if there are some oranges still fresh.
- The following is a sample of the output format.

Output
en(0, 0) en(0, 3) en(1, 3) en(2, 3) en(-1, -1) de(0, 0) en(1, 0) en(0, 1) de(0, 3) en(0, 4) de(1, 3) en(1, 2) en(1, 4) de(2, 3) en(2, 4) de(-1, -1) en(-1, -1) de(1, 0) en(2, 0) de(0, 1) de(0, 4) de(1, 2) de(1, 4) de(2, 4) de(-1, -1) en(-1, -1) de(2, 0) de(-1, -1) 2 1

3. Submission

To submit your files electronically, login DomJudge website through the following url:

https://108-1-ds-judge.ate.cs.ccu.edu.tw/

Press the submit button and choose the homework questions you want to submit. After submitting your code, DomJudge will give you a result to tell you whether your code is correct

or not. However, during the demo time, your code will be evaluated by different sets of test cases. Please make sure your code can work correctly based on the description above. Additionally, you must compress your code and the README file into a **zip** file and upload it to Ecourse2.

4. Grading policies

- 10%- Readme file, code style, and comments in the source code
- To keep source code maintainable and readable, you should **add English comments to your source code**. For this assignment, please also compose a small "**README.txt**" which contains a **brief** explanation of **how to compile your program** and **what problem you met**.
- 45%- Implement the "validation and evaluation of expressions", if you do not use stacks, you will get 0%.
- 45%- Implement the "Minimum time required to rot the maximum number of the oranges", if you do not use queues, you will get 0%.

Notice that if your homework is copied from your classmate, **you** and **your classmate** will get **0%** in this homework!