Advanced Algorithms and Data Structures
Stacks and Queues
Exercises

Prepared by LY Rottana

Stacks

1. By using stack, write a method which will read in a string that will return true if it contains a valid parenthesis pattern and false otherwise.
2. The same as previous exercise, but the string now contains two types of encloses: parenthesis and brackets.
3. Write a method to determine if a string consists of number of A's followed by an equal number of B's. (By using stack, not using the count of A & B).
4. Write a method to determine if a string consists of number of A's followed by an equal number of B's followed by an equal number of C's.
5. Write a method that using stack to calculate the postfix expression.
   Example:

Postfix Expression	Result
1 2 +	3
1 2 3 * + 4 +	11
8 5 * 7 4 2 + * +	82
6 8 2 / 1 - *	18

*** Postfix Expression**

A postfix expression: operators follow their operands; for example, to add 3 and 4, we write "3 4 +" rather than "3 + 4".

If there are multiple operations, the operator is given immediately after its last operand so the expression written "3 – 4 + 5" would be written "3 4 – 5 +" in postfix form.

Queues

6. Write a program to compute average waiting time at a bus stop. Each line of input contains 3 items: a code, a time (hour : minute : second), and an integer. The code is either 'b' in which case the input represents the arrival of a bus at the stop at a specified time, or 'p' in which case the input represents the arrival of a group of people at the stop at the specified time. The integer represents the number of people in the group in the case of 'p' code, and the number of people for which the bus has room for in the case of a 'b' code. The people are assumed to form a single line, entering the line at the rear and leaving the line at the front to get on the bus. If a bus arrives with room for n people, the first n people on the line get on the bus. If there are less than n people on the line, they all get on the bus, and the bus does not wait for more people to arrive. The input lines are to be in order of arrival times. Each time that a 'b' code is read, print the data and on the next line print:

```
Time H:MN:S. XX people get on the bus, YY people now remain.
```

Each time that a ‘p’ code is read, print the data and on the next line print:

```
Time H:MN:S. XX people arrive. YY people are now on the line.
```

When no inputs remain, print the average waiting time:

```
The average waiting time is ZZ seconds.
```

Test your implementation with the following scenario:
- at 07:30:00, 10 people arrive at the bus stop
- at 08:05:30, 15 people arrive at the bus stop
- at 08:20:45, a bus with 18 available seats arrives
- at 09:00:25, 22 people arrive at the bus stop
- at 09:10:55, a bus with 20 available seats arrives
- at 10:05:00, 5 people arrive at the bus stop
- at 10:30:15, a bus with 30 available seats arrives

7. Modify the program of the previous exercise so that there are two lines of people waiting for the bus. One line enters at the front of the bus, the other at the back of the bus. When a group of people arrive at the bus stop, they go to the back of the shorter line. When n people enter the bus, half of them enter from the front and half of them from the back. If n is odd, one more person enters from the front than from the back. If one line becomes empty before n people have boarded, the remaining passengers are taken from the other line.

8. Implement a queue using two stacks. Create a new class **QueueFrom2Stacks** for the implementation and another class **TestQueueFrom2Stacks** for testing.