METROPOLITAN STATE UNIVERSITY

ICS 240 - 02: Introduction to Data Structures
Fall 2018

Assignment 5: Exercies on the Stack Data Structure
Out: Friday, October, 19th, 2018
Due: Friday, November, 2nd, 2018

Total points: 50

Submission - Important

This assignment is divided into two parts. The first part includes paper-pencil-based exercises, while the second part includes programming exercises.

Part 1: Submit a **HARD COPY** of Part 1 questions in class on November, 2nd. **No electronic submissions will be accepted** for part 1.

Part 2: Solutions to Part 2 questions are to be uploaded on the designated D2L folder, by 11:59 PM on November, 2nd**.**

Part 1: Classical Algorithms that uses a Stack

Write your answers to the following questions **neatly**

Exercise 1.1 (5 Points): For each of the following expressions, show how to use a stack to check whether the expression is balanced or no. Show the stack after each `push` and `pop` operation.

1.1.1) ([ ] { }) ( )))

1.1.2) ({ } ([[ ]]) ( ))

Exercise 1.2 (15 Points): Answer parts a), b), and c) for each of the given **infix** arithmetic expressions. Neatly show the stack(s) after each `pop` and `push` operation.

a) use stack(s) to evaluate the infix arithmetic expression
b) use stack(s) to convert the infix expression to postfix expression
c) use stack(s) to evaluate the postfix expression.

1.2.1) `8 + 3 * 5 - 2 + 9`
1.2.2) `6 + ( 4 * 5 - 9/3 + ( 8 - 2) )`

Exercise 1.3 (5 Points): Suppose that an intermixed sequence of push and pop operations are performed. The pushes push the integers 0 through 9 in order and the popped item is printed out. Write down the sequence of push and pop operations to print the following sequences. Make sure the integers are pushed in order from 0 to 9 while the pop operations can be placed anywhere between the push operations.

1.3.1) `4 3 2 1 0 9 8 7 6 5`
1.3.2) `4 6 8 7 5 3 2 9 1 0`

Part 2: Programming Exercises

Implement the following static methods that uses `IntStackInterface` as the input data type. All the methods must satisfy the following:

- Input parameters **MUST** be on type `IntStackInterface`. However, local variables can be either `IntArrayStack` or `IntLinkedStack`. **Do NOT use the Stack provided by the Java library.**
- All the methods are static and are implemented in the `StackDriver` class.
- The input stacks should remain unchanged except only as required by the method.
- Include appropriate comments in your methods to explain your algorithm.

1- `stackToInt`: a method that takes a stack as a parameter. Assume that the input stack contains 7 or less integers all of them in the range 0 to 9. The method returns as output an integer number representation of the values stored on the stack where the most significant digit is at the top of the stack. For example, if the input stack is as shown, the output integer is 5478.

2- `flip`: a method that takes a stack as input and returns as output another stack that has the exact same values as the inputs stack but in reverse order. The input stack must remain unchanged.

3- `popBottom`: a method that takes a stack as parameter and returns as output the value at the bottom of the stack and removes that bottom element from the input stack. All other elements in the stack should remain unchanged.

4- `popSome`: a method that two parameters, a stack `s` and an integer value `count`. The method then pops `count` values from the stack. The method produces as output the sum of the popped values. If the stack has less than `count` values, the method returns -1.

5- `extractFromStack`: a method that takes two input parameters, a stack and an integer value. The method removes all occurrences of the input value from the input stack. All other elements in the stack must remain unchanged.

6- `equalStacks`: a method that two stacks as input parameters, and return true or false based on whether the two stacks are equal or no. Two stacks are equal if they have the same values in the same order. The two input stack must remain unchanged after the call to this method.

Submission - Important

For this question, you need to upload only one file, named `StackDriver.java`. This file should include the implementation of the five methods explained above AND a `main` method that includes one test case for each one of the 6 methods. Note that, I will test your driver using `IntStackInterface`, `IntArrayStack`, and `IntLinkedStack` that are posted on D2L so make sure your driver runs without errors.