Lab 02 Specification – Implementing Stack ADT and few of its applications Due Friday, 02 Feb 2018 1PM (after 1PM submission will not be accepted)

50 points

Lab Goals

- Proposing solutions to highly regarded algorithmic problems.
- Implementing Java programming solutions.

Assignment Details

This lab exercise is going to be a team based lab. So you are allowed to collaborate with your team members and do the lab work together as a team. You are expected to submit your lab work on or before the due date by sending an email to the Instructor. See more details about submission in the last section.

Preliminary Requirement:

1. You should implement the task using Java programming.
2. You should create a folder called cmpsc250-lab02-YOURFirstInitialLastName. So for example cmpsc250-lab02-amohan
3. You should create two sub folders namely: src and classes. All your source code ”.java” needs to be inside the src folder. All your classes files ”.class” needs to be in your classes folder.

   A simple technique to do this organization is by using the following command during compilation:
   javac -d ../classes -cp . *.*
4. We adhere to the group assignment that was done in our earlier class and the group sheet is provided in the course webpage. This lab work will be done within your group and all the team members are going to contribute to the lab in a fair manner.

If you fail to adhere to the requirements, your lab work will not be evaluated and thereby you and your team will not receive any points for the lab.

The Stock Span Problem

Problem Definition: The span $S_i$ of a stock’s price on a certain day i is the maximum number of consecutive days (up to the current day) the price of the stock has been less than or equal to its price on day i. How to efficiently find $S_i$ for any given day i?

Approach: We discussed this in detail during our lecture. There is pseudocode provided to you in our lecture slides. You are allowed to incorporate the pseudocode into your implementation.

Requirement:

1. You should create a class called StockSpanProblem and provide the implementation of your algorithmic solution to the problem above.
2. You are required to implement the efficient solution that executes linear.

3. You need to use the custom Stack ADT that we discussed during lecture session. You can either reuse the code from our lecture slides folder or recreate your own Stack ADT from scratch. But you cannot use the Java builtin Stack ADT.

4. Input: The input for the number of days, stock price for each day needs to be user provided. The input can either be provided through console input or your program can read and parse a text file that contains the stock price for each day. The elegant way of doing would be the second approach, so you can run your program for many number of days ("Let us say for 1 year"). Approach 1 is too time consuming and burden the user for larger number of days. But, it is totally upto you on how to design the input scheme.

5. Output: The output needs to be displayed by using two display options:
   1) All days: Span for all days needs to be displayed along with the day number and the stock price for the day.
   2) Individual day: Span for a particular day (user provided) needs to be displayed along with the stock price for the day.
   Display options needs to be triggered by the user through keyboard inputs.

The Towers of Hanoi Problem

Problem Definition: In the classic problem of the Towers of Hanoi, you have 3 rods and N disks of different sizes which can slide onto any tower The puzzle starts with disks sorted in ascending order of size from top to bottom (e.g., each disk sits on top of an even larger one) You have the following constraints:

1. Only one disk can be moved at a time.
2. A disk is slid off the top of one rod onto the next rod.
3. A disk can only be placed on top of a larger disk.

Approach: You can use recursive approach to solve this problem effectively. You need to consider proper recursive cases how and where to do the push and pop operations.

Requirement:

1. You should create a class called TowerOfHanoi and provide the implementation of your algorithmic solution to the problem above.
2. Your implementation need to move the disks from the first tower to the last tower using Stacks. You are allowed to consider each tower, as a stack in your implementation.
3. You need to use the custom Stack ADT that we discussed during lecture session. You can either reuse the code from our lecture slides folder or recreate your own Stack ADT from scratch. But you cannot use the Java builtin Stack ADT.
4. Input: The input for the number of disks, the name/id of the disk, and the name of the 3 towers needs to be user provided.
5. Output: The output needs to be displayed by using three display options:
   1) initial list: show the initial number of disks and the name/id of the disks in the first tower.
   2) intermediate steps: show all the steps taken in moving the disks from the source to destination tower. So for example, if you have a step that moves disk D3 from tower A to tower C, then your program needs to print the following to the output console:
      A → C (D3).
   3) final list: show the final number of disks and the name/id of the disks in the last tower.
   Display options needs to be triggered by the user through keyboard inputs.
Sorting numbers using Stack

**Problem Definition:** Sort a stack of random numbers in ascending order. You should not make any assumptions about how the stack is implemented. The following are the only functions that should be used to write this program: push — pop — peek — isEmpty

**Approach:** You should think of using more than one stack to implement this problem. But at the same time, you need to use the least number of stacks for your implementation.

**Requirement:**

1. You should create a class called StackSorting and provide the implementation of your algorithmic solution to the problem above.
2. Your implementation need to use Stack for holding initial and final data.
3. You need to use the custom Stack ADT that we discussed during lecture session. You can either reuse the code from our lecture slides folder or recreate your own Stack ADT from scratch. But you cannot use the Java builtin Stack ADT.
4. Input: The input for the total number of elements in the stack needs to be user provided. The values that corresponds to each element in the stack needs to be randomly generated. The upper limit to the size of the stack cannot be above Integer Max in Java.
5. Output: The output needs to be displayed by using two display options: 1) initial list: show the initial unsorted list of elements and their values in the stack. 2) final list: show the final sorted list of elements and their values in the stack. Although not required, the elegant way of displaying the output is by writing the output to a file (comma seperated or new line seperated).

Display options needs to be triggered by the user through keyboard inputs.

**Submission Details**

1. You are required to submit this lab by sending an email with a zipped version of your cmpsc250-lab02-YOURFirstInitialLastName folder.
2. Subject of your email should say "CMPSC250: Team X Lab 02 Submission". Here X needs to be replaced with your group number. Refer the group sheet in the course webpage.
3. One email should be sent for your teams lab submission.
4. Send the email to amohan@allegheny.edu // CC the email to all your team members.
5. You should add the following statement in the body of your email
   By doing this submission, I understand that I and my team members are subject to the Honor Code policy.
   Lab submitted by: X1, X2, and X3 (Here X1, X2, and X3 are the name of your team members)
6. Shifting team members is not ideal and not allowed, unless there is an extreme situation which is discussed with the instructor prior to your submission.