Lab 01 Specification – Explore Stack ADT
Due (via your git repo) no later than 8 a.m., Thursday, 31 January 2019.
50 points
[An individual lab assignment]

Lab Goals

- Quick review of GitHub and git commands.
- Implementing Stack ADT and few of its applications.
- Translating Algorithm to Program
- Visualizing graphs by using StdDraw in Java to plot experimental results

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`
- Often times, the lab specification is purposefully developed to include open ended tasks and not exposing too low level details. The intention is to trigger you towards further refining your thinking skills. Thinking is more important to solve any algorithmic problem provided. Thinking through, may lead you to ask further clarifying questions, which is what my expectation is. So, there will be one or more open ended parts added to the lab. Attending labs regularly and steering discussions with the Professor, TA's and your classmates is going to be the success mantra for doing a great job in the labs. Waiting till the last minute, will restrict a student from both learning the content discussed and to have a negative effect to the effectiveness of the lab work produced.

Learning Assignment

If you have not done so already, please read all of the relevant "GitHub Guides", available at https://guides.github.com/, that explain how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as "Mastering Markdown" and "Documenting Your Projects on GitHub"; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- Sedgwick chapter 01, section (1.1 - 1.3)

Assignment Details

Firstly practicing how to transform algorithm into program written in Java. Secondly, get to know how to time the program execution through the program running time. Thirdly, using builtin functionality in Java to show experimental results on an applet.

The Time Series 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 had discussed two approaches in detail during our lecture session. There is pseudocode provided in our lecture slides. It is acceptable, to incorporate the pseudocode or reuse source code from lecture folder into your implementation.

Requirement:

1. It is required to do the implementation using Java programming language,
2. Create a separate java file called TimeSeries01.java and provide the implementation of your inefficient algorithmic solution to the problem above using a function named **ComputeSpan**. We had discussed some source code for this part during our lectures, so it is acceptable to reuse the code from lecture discussions.
3. Create a separate java file called TimeSeries02.java and provide the implementation of your efficient algorithmic solution to the problem above using a function named **ComputeSpan**. We had discussed the algorithmic pseudocode for this part during our lectures, so it is required to translate the pseudocode to Java source code.
4. It is required to use the custom Stack ADT that we had discussed during our lecture session. You can either reuse the code directly from our lecture slides folder or recreate your own Stack ADT from scratch. But it is not acceptable to use the Java builtin Stack ADT. There will be points deducted if the builtin Stack ADT is used in student submission.
5. Input: The input for the number of days, stock price for each day needs to be user provided. The input should be provided through external file inputs such as file01.txt, file02.txt, file03.txt, file04.txt, file05.txt. It is required for your program to generate these datasets, with a random price value from [1-100]. Later your program should read and parse those text files that contains the stock prices for each day and build the input arrays based on that. It is recommended to have your program run on input dataset sizes 365, 730, 1460, 2920, 5840 days. As you may note, the dataset sizes are incremental by one year and can also be represented as 1, 2, 3, 4, 5 years. So, while generating these datasets, it is important to make sure the number of days for each of the files are as shown below:
   - "file1.txt" $\rightarrow$ 365 days
   - "file2.txt" $\rightarrow$ 730 days
   - "file3.txt" $\rightarrow$ 1460 days
   - "file4.txt" $\rightarrow$ 2920 days
   - "file5.txt" $\rightarrow$ 5840 days
6. Output: The output is required to be written to two folders namely "out-timeseries01" and "out-timeseries02" inside your lab repository. Executing the ComputeSpan function in TimeSeries01 and TimeSeries02 classes, should create output files called "out1.txt", "out2.txt", "out3.txt", "out4.txt", "out5.txt" respectively. The output files should contain the span value for each day given in the input datasets.

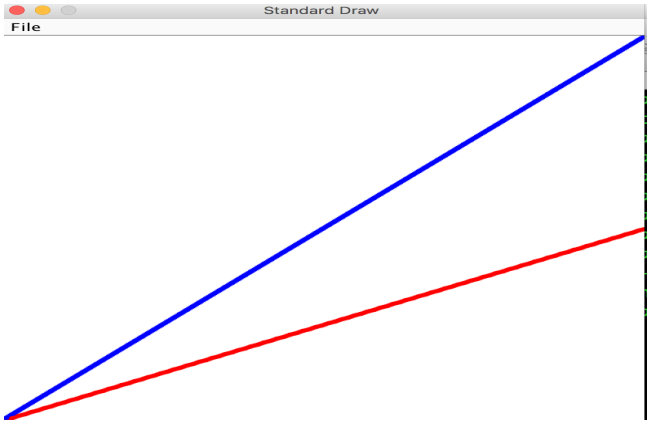
Experimental Study

Write a function called AlgorithmTimer that executes the ComputeSpan function in TimeSeries01 and TimeSeries02 classes. The ComputeSpan functions should be executed on different datasets and the program is required to clock the running time for different datasets.

The function should then use **JavaDraw** program available for you in the lab specification attachment. This program is primarly used to plot the running time for different datasets. The instructions for executing the program is provided below:
Note: X axis = Dataset ID [1, 2, ... 5] Y axis = Running time of your algorithm. There is two lines expected to be shown in your graph, one for TimeSeries01 (red line) and another for TimeSeries02 (blue line). You may refer to the TestStdDraw class for additional details on how to show the lines in the graph with different colors. It is the general intuition that based on our analysis of the TimeSeries algorithm, that the blue line should be lower than the red line in the graph, based on the time efficiency.
How to run the applet program JavaDraw and see the results?

- cd into the JavaDraw/src directory in the terminal
- Compile the code by typing in**javac *.java -d classes**
- Run the program by first cd into the JavaDraw/classes directory. Then by typing in **java TestStdDraw**

  After running the program, an applet should open up like below:



Submission Details

For this assignment, please submit the following to your GiHub repository by using the link shared to you by the course instructor:

1. Commented source code from the “TimeSeries01” program.
2. Commented source code from the “TimeSeries02” program.
3. Commented source code from the “AlgorithmTimer” program.
4. A readme file explaining how to compile and run your program. In addition, include a detailed self reflection of the lab exercise in the readme file. The self reflection may include answers to questions such as:
   - ”What challenges did you face in this lab?”
   - ”What did you like in this lab?”
   - ”What did you not like in this lab?”
5. It is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus. Make sure to add the statement ”This work is mine, unless otherwise cited.” in all your deliverables such as source code and other textual documents.

Grading Rubric

1. The Time Series Problem is worth 30 points; 10 points will be awarded for TimeSeries01 and 20 points will be awarded for TimeSeries02. There will be points taken out, if the dataset creation procedure is not included correctly in your program.
2. The Experimental Study is worth 20 points; I am looking for AlgorithmTimer to produce the graph with two lines, as outlined in the requirement section above.
3. There will be full points awarded for the lab, if all the requirements in the lab specification are correctly implemented. Partial credits will be awarded if deemed appropriate.
4. Failure to upload the lab assignment code to your git repo, will lead you to receive ”0” points given for the lab submission. In this case, there is no solid base to grade the work.
5. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in github. In this way, an updated version of student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then in this case, there is 0 points to the student automatically for the lab work.
6. If you needed any clarification on your lab grade, talk to the instructor. The lab grade may be changed if deemed appropriate.