Lab 02 Specification – A Hand-on Exercise to practice Algorithm Implementation
50 points

Due by: 03/18/2021 2:00 PM

Lab Goals

- Sharpening your thinking skills to solve ADT challenges.
- Practice implementing algorithms in a programming language of your choice (Python or Java).

Learning Assignment

If not done previously, it is strongly recommended to read all of the relevant "GitHub Guides", available at the following website:
`https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading assignment is useful to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, it is also recommended to do the reading assignment from the section of the course textbook outlined below:

- **Sedgewick chapter 01, section 1.3 Stacks**

Assignment Details

Now that we have discussed some basics of algorithms and analyzed algorithms with data structures together in the last few lectures, it is now time to do it practically. In this lab, students will practice developing a variety of algorithms to retain the knowledge from the class discussions so far. This also includes developing one or more code files to implement a series of algorithms using a programming language such as Python or Java. At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL(s) to clarify if there is any confusion related to the lab and/or class materials.

The Professor proof-read the document more than once, if there is an error in the document, it will be much appreciated if student(s) can communicate that to the Professor. The class will be then informed as soon as possible regarding the error in the document. Additionally, it is highly recommended that students will reach out to the Professor in advance of the lab submission with any questions. Waiting till the last minute will minimize the student chances to get proper assistance from the Professor and the Technical Leader(s).

Students are recommended to get started with this part in the laboratory session, by discussing ideas and clarifying with the Professor and the Technical Leader(s). It is acceptable to discuss high-level ideas with your peers, while all the work should be done individually. Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

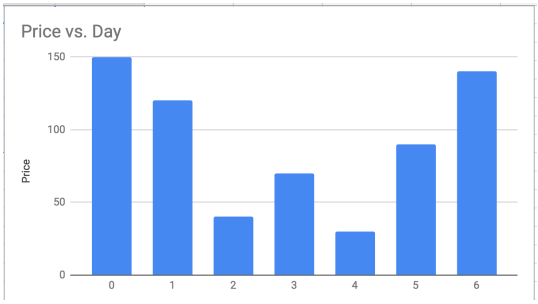
1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.
2. Submitting a copy of the other's program(s) and algorithm(s) is strictly not allowed. Please note that all work done during lab sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as skill test, exams, etc $\cdots$

Part 01 - A Time Series problem implementation (15 points)

Implement the Time Series problem using a series of requirements outlined below.

1. The problem definition is provided below:

   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. Let us suppose we are given with the stock prices for different days. Identify the span S for those days.



   Span[0,1,2,3,4,5,6] = {1,1,1,2,1,4,6}

2. Please refer to Week03 slides and your notes for more details from class discussion related to this problem.

3. In order to easily access the Time Series algorithm Version 1 is provided below:

   Algorithm - ComputeSpan(P)

   Input: an n-element array P of numbers such that P[i] is the price of the stock on a day i.

   Output: an n-element array S of numbers such that S[i] is the span of the stock on a day i.

```
for i = 0 to n do
  h = 0, done = false
  repeat
    if P[i−h] <= P[i] then
      h = h+1
    else
      done
  until (h = i) or done
  S[i] = h
return S
```

4. The starter code file is provided inside the lab repository in the file(s) named, `TSA.java` and `TSA.py`. Here TSA refers to Time Series Algorithm.

5. Implement the **computeSpan** method using the algorithm outlined in the previous point. One challenge in this part is to understand how to implement the repeat until block in your program. We discussed this challenge in class. Please refer to your class notes. The idea is to use a While loop, with conditions opposite to the one outlined in the algorithm using until.
6. For more details on the requirements for this program, please look at the comments in the source code file.
7. To validate the correctness of the output of the program, please pick an example from the slides and/or your notes and validate the correctness of the program.

Part 02 - 9 Balls Puzzle (20 points)

The problem definition is provided below:

We have a total of 9 balls, that are identical in appearance. Out of these 9 balls, 8 are of the same weight and 1 is heavier. Identify this heavier ball by using a balance twice. Assume that we are given a balance scale that can be used to measure one or more balls at any given time. That is, we can load one or more balls on both sides of the balance and measure their weight. An important constraint attached is this problem is that we can only use the balance up to two times to solve this problem.

1. The starter code is provided inside the lab repository in the file(s) named, `Balance.py` and `Balance.java`
2. The starter code has a method called generate which is invoked at the start of the program execution. This method generate a list of ball weights. We use a random value generator to assign random values and make sure that all values are same except one.
3. The balance method can be used at any point of time to measure the total weight of one or more ball(s). This method accepts the list of balls (based on their weights), first and second to represent the balls to be weighed on the left and right respectively.

   The first and second parameter is expected to be of type dictionary. The dictionary should have the start and end key and their values. For example:

   Python:

   ```
   first = {"start":0,"end":4}
   second = {"start":5,"end":8}
   ```

   Java:

   ```
   Map<String, Integer> first = new HashMap<String, Integer>();
   first.put("start", 0);
   first.put("end", 4);

   Map<String, String> second = new HashMap<String, Integer>();
   second.put("start", 5);
   second.put("end", 8);
   ```

The output is **0** (if the weights are equal on both sides), is **-1** (if the weight of the ball(s) on the left is greater than the weight of the balls on the right.), and is **+1** (if the weight of the ball(s) on the right is greater than the weight of the balls on the left.)

The balance method can be called only twice. The program utilizes the exit() method to automatically exit the program if the number of times the method called reaches two. The variables calls and limit are used in the starter code to implement this restriction.

4. Implement the **puzzle** method to identify the heavier ball. This method is currently incomplete and the task is to complete this method. Include the appropriate lines of code and the proper logic to call the balance method a maximum of two times. This method is expected to output the heavier ball weight. An important hint to solve this problem is provided below. First solve this problem in paper and plan your solution before you start coding. Here focus is on solving rather coding. Coding process is secondary and starts only after solving the problem. The hint is purposefully left at a high level & open-ended, to encourage students to think and solve this problem and have fun!

Hint: Don't restrict yourself to divide the balls just into two groups ...

Part 03 - To Solve (10 points)

An important part of integrating ADT into an algorithm is to understand how the ADT works mathematically. The axioms to solve are provided in the `axioms.md` file. Answer the questions listed in the file. The solution to the questions listed should be provided as instructed in the file.

Part 04 - To Think (5 points)

Another important part of algorithm development is to develop thinking skills. By now, we have developed one variation of the defective coin problem. The key point in solving such a problem is thinking through and solving the smaller subproblems and later aiming to solve the bigger problem. Think and come up with an idea to propose a different algorithm where such a computation (that is, to find an element that is different from the other similar elements) is useful. The new idea should enrich and enhance what had been developed already. Include a summary of one or more ideas in a file named **ideas.md**.

Part 05 - Honor Code

Make sure to **Sign** the following statement in the `honor-code.txt` file in your repository. To sign your name, simply replace Student Name with your name. The lab work will not be graded unless the honor code file is signed by you.

This work is mine unless otherwise cited - Student Name

PS next page ...

Submission Details

For this assignment, please submit the following to your GitHub repository by using the link shared to you by the Professor:

1. Commented source code in either Python “TSA.py” or Java “TSA.java”program.
2. Commented source code in either Python “Balance.py” or Java “Balance.java”program.
3. A document containing the solution to the questions on ADT axioms, in a file named `axioms.md`.
4. A document containing the ideas to enrich the algorithm developed to solve the diurnal range, in a file named `ideas.md`.
5. A signed honor code file, named `honor-code.txt`.
6. To reiterate, 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.

Grading Rubric

1. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits may be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your GitHub repository will lead to receiving no points given for the lab submission. In this case, there is no solid base to grade the work.
3. 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 the 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 the 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 are no points awarded to the student.
4. If a student needs any clarification on their lab grade, it is strongly recommended to talk to the Professor. The lab grade may be changed if deemed appropriate.