Lab 01 Specification – A Hand-on Exercise to practice Algorithm Development
50 points

Lab Goals

- Quick review of GitHub and git commands.
- Refresh your ability to write algorithms.
- Think about how to solve algorithmic challenges.
- Learn algorithm implementation using a programming language such as Java.

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`

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**

Assignment Details

Now that we have discussed some basics of algorithms and analyzed multiple algorithms 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 Java.

The lab has two parts. The first part is required to be completed by the end of the lab session. The second part is required to be completed within a one week time period. Please make sure to push your code file(s) and commit by the deadline provided in each of the following section(s).

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) 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 tests, exams, etc.

At any duration during and/or after the lab, students are recommended to team up with the Professor and the Technical Leader(s) to clarify if there is any confusion related to the lab and/or class materials.

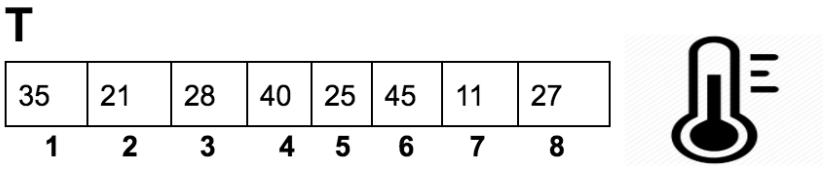
Section 1: In-Lab Exercise (20 points)

Attendance is in general a required component in computer science class(es) and laboratory session(s). By attending these sessions regularly, students get an opportunity to interact with the Professor and the Technical Leaders(s). This interaction helps maximize the student's learning experience. Attendance is taken in the form of Mastery Quizze(s) and by checking the commit log of the student's In-Lab exercise submission. No student is allowed to take the Mastery Quizze(s) outside the laboratory room unless prior arrangements are done with the Professor. It is against the class honor code to violate the policy outlined above.

Part 01 - Attendance Mastery Quiz (5 points)

The Mastery Quiz is graded based on a Credit/No-credit basis. Students are required to complete the Mastery Quiz only in the laboratory room, which is in **Alden 109** before 3:15 PM. Submissions after 3:15 PM is not acceptable. Students who had completed the Mastery Quiz according to the policy outlined above will automatically receive (5) points towards their lab grade. The link to the Mastery Quiz is provided below:
`https://forms.gle/oaqWYRjDxcRvb4tKA`

Part 02 - A Simple Exercise to develop an algorithm formally (15 points)



Students are required to complete this part only in the laboratory room, which is in **Alden 109**. It is required to complete the final commit by the end of the laboratory session, which is before **4:20 PM** on the same day that the lab is given to the students. There is late submission accepted for this part, according to the late policy outlined in the course syllabus.

A diurnal range is scientifically defined as the difference between the maximum and minimum temperature from the set of recordings taken in a day. There are several variations to define this. As per scientific articles referenced below, a large diurnal range can show a positive effect in Wine production. There are many wineries in the Erie area and hence it is great to look at this problem and solve it algorithmically. Develop an Algorithm (formally) to compute the diurnal range by using a set of temperature recordings for any particular day.
`https://www.decanter.com/learn/what-is-diurnal-range-ask-decanter-413231/`

1. For simplicity, a starter code with a file named `diurnal-algorithm.tex` is provided in the lab repository. The starter code has an example formal algorithm discussed and practiced during class discussions.
2. A latex file may be developed using the Overleaf website link provided below:

   `https://www.overleaf.com/login`
3. Please note: The Overleaf website may prompt to register and log in before providing the options to compile latex files and generate pdf files. This should be a straight forward process. If there are any questions, students are encouraged to ask the Technical Leader(s) and the Professor.
4. An alternative approach is to install latex on your laptop using the web resources below:
   (a) **MAC**: `https://www.latex-project.org/get/`
   (b) **Ubuntu**: `https://milq.github.io/install-latex-ubuntu-debian/`
   (c) **Windows**: `https://www.youtube.com/watch?v=oI8W4MvFo1M`
5. An easier way to do Latex development is through Overleaf. To do the development without the internet, it is best to install latex on your laptop.
6. There are also apparently options to install latex as a plugin in popular text editors such as Atom. This is not something that had been tried out by the Professor yet. Hence, it is up to the students to try out this option.
7. **ADD** the following statement to all your submission files including the file named `diurnal-algorithm.tex`.

   This work is mine unless otherwise cited - Student Name
8. Make necessary modification(s) to the file named `diurnal-algorithm.tex` to include the steps to compute the diurnal range. The computation should include steps such as finding the minimum, finding the maximum, and finding the difference between the minimum and maximum temperature recording.
9. The algorithm should be developed in a formal manner using a similar style as we did in the class examples. Please refer back to the lecture slides and your class notes to look at the examples discussed in class.
10. It may be too tempting to propose a solution that uses two separate for loops (not nested). However, it is expected that the algorithmic solution submitted should contain only one for-loop.
11. **Hint:** One way of doing this is to start from the first value in the set of temperature recordings and compare the first value with the other values within the loop to identify the minimum and the maximum value. In this approach, the assumption is that the algorithm is restricted to only the values in the temperature recordings set. For example, there is no external access to a minimum and maximum values such as **Integer.MIN_VALUE** and **Integer.MAX_VALUE**. It is acceptable to relax this assumption in a student's submission for this part.
12. It is required as part of this lab submission for students to compile the latex files and generate a PDF version of the file. The PDF file should be named as `diurnal-algorithm.pdf` and uploaded to the repository. Both the **tex** and **pdf** files will be used for grading purposes.

Part 03 - Pair Discussion (5 points)

Students are required to complete this part only in the laboratory room, which is in **Alden 109**. It is required to complete the final commit by the end of the laboratory session, which is before **4:20 PM** on the same day that the lab is given to the students. There is no late submission accepted for this part unless prior arrangements are done with the Professor.

1. Pair up with at least one of your peers.
2. Talk to your peer and discuss at least one application, in which, computing the difference between the maximum and minimum from a set of values matters.
3. Please note, we already looked at one such application in the diurnal range. The pair should identify a different application where finding such a difference is useful.
4. Each person in the pair should discuss their own idea and write a short summary to explain their idea. The other person should critic the idea of their peer by providing them with at least one strong and weak point related to their peer's idea.
5. The summary should also include the original idea and the critic points provided by the peer.
6. Include the name of your peer in the summary file.
7. The pair should additionally discuss other ideas to solve the problem in an Asymptotic linear time, while still keeping the solution within the one for loop.
8. Each person in the pair should then include their discussion points in the summary document. If the team is unable to come up with an alternative solution, then this should be indicated as the discussion results.
9. Name the summary file as **pd-summary.pdf**.
10. Here **pd** refers to pair discussion. In order to pass the submission requirements, the file should be generated using a markdown (or) PDF format. **ADD** the following statement to all your code files including the file named `pd-summary.pdf`.

This work is mine unless otherwise cited - Student Name

After completing all the parts in Section 01, it is acceptable for students to start working on Section 2 remotely. Although it is strongly recommended to be in the laboratory room till the end of the lab session, and work on the lab along with other colleagues.

Section 2: Take Home Exercise (25 points)

Students are recommended to get started with this part in the laboratory room, by discussing ideas and clarifying with the Professor and the Technical Leader(s). All the part(s) in this section is due in **one week** time. It is acceptable to discuss high-level ideas with your peers, while all the work should be done individually. The deadline for submitting the part(s) in this section is **31st Jan 2020, 8:00 AM**. Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

Part 01 - An Expense Tracker Algorithm (10 points)

Develop an algorithm to create an Expense Tracker using a series of requirements outlined below.

1. The starter code is provided inside the lab repository in a file named, `exp-tracker1.tex` and `exp-tracker1.pdf`.
2. First, analyze the algorithm provided in the starter code. Identify the asymptotic running time of the algorithm provided.
3. The algorithm takes in a set of expenses made by a person every day in a calendar month. The algorithm then finds the daily average for the expenses done for each day in the month.
4. The algorithm provided in the starter code is not generally classified as the most efficient solution. Think about ways to solve this problem efficiently.
5. The algorithm in `exp-tracker1.tex` is also shown below to easily access it.

   Algorithm: ExpenseTracker(E)

   Input: An n-element array E of expenses each day.

   Output: An n-element array A of values such that A[i] is the average of elements E[0], E[1], ... , E[i]

   1: Initialize $a, i, j = 0$
   2: **for** $(i = 0; i < n; i = i + 1)$ **do**
   3: $\quad a \leftarrow 0$
   4: $\quad$ **for** $(j = 0; j < i; i = i + 1)$ **do**
   5: $\quad\quad a \leftarrow a + E[j]$
   6: $\quad$ **end for**
   7: $\quad A[i] \leftarrow a/(i + 1)$
   8: **end for**
   9: **return A**

6. The efficient solution should solve this problem in linear time. The details are not provided so as to make sure that the students are able to develop their own thinking skills.
7. Create a new file named `exp-tracker2.tex` and include the formal algorithm of an efficient solution.
8. **ADD** the following statement to all your code files including the file named `exp-tracker2.tex` .

   This work is mine unless otherwise cited - Student Name

9. It is required as part of this lab submission for students to compile the latex files and generate a PDF version of the file. The PDF file should be named as `exp-tracker2.pdf` and uploaded to the repository. Both the **tex** and **pdf** files will be used for grading purposes.

Part 02 - An implementation of the Defective Coin Problem (10 points)

Write a Java program that implements a `Defective Coin Detection` program using a series of requirements outlined below.

1. The starter code files are provided inside the lab repository in files named, `DCD1.java` and `DCD2.java`. Here DCD refers to Defective Coin Detection.

2. **ADD** the following statement to all your code files including the files named `DCD1.java` and `DCD2.java`.

   This work is mine unless otherwise cited - Student Name

3. First, analyze the brute force algorithm that we discussed in class. Identify the asymptotic running time of this program. The brute force algorithm is provided below so as to easily access it.

   Algorithm - Find Defective Coin (W, n)
   Input - A set of coin weights associated with a collection of coins.
   Output - The position of the defective coin.

   ```
   for i = 0 to n − 1 do
       if W[i] <> W[i + 1] then
           if W[i] < W[i + 1] then
               return i
           else
               return i + 1
           end if
       end if
   end for
   ```

4. Implement the algorithm by making necessary code modifications to the file named `DCD1.java`. The code file has a starter code with detailed comments listed to describe the current behavior of the code.

5. Next, analyze the divide and conquer algorithm that we discussed in class. Identify the asymptotic running time of this program. The divide and conquer algorithm is provided in the next page so as to easily access it.

Algorithm - Find Defective Coin ($W, low, high$)
Input - A set of coin weights associated with a collection of coins.
Output - The position of the defective coin.

```
1: first ← SCALE(low to (low + high)/2);
2: second ← SCALE((low + high)/2 to high);
3: if (high − low) is equal to 1 then
4:     if first < second then
5:         return low;
6:     else
7:         return high;
8:     end if
9: else
10:    if first < second then
11:        return FDC(W, low, (low + high)/2);
12:    else
13:        return FDC(W, (low + high)/2, high);
14:    end if
15: end if
```

6. Implement the algorithm by making necessary code modifications to the file named `DCD2.java`. The code file has a starter code with detailed comments listed to describe the current behavior of the code.

7. Both programs should output the defective coin, which is the coin that is of lesser weight from the set (array) of coin weights given.

Part 03 - To Think (5 points)

An important part of algorithm implementation is to develop thinking skills. By now, we had implemented two interesting solutions connected to the Defective Coin problem. Think and come up with ideas to extend the program developed in the earlier parts of this section. The ideas should enrich and enhance what had been implemented already. For example: is it possible to solve this problem differently? or more efficiently? If not think and propose a different variation of this problem in a different setting. Include a summary of one or more ideas in a file named **ideas.pdf**. In order to pass the requirements, the file should be generated using a Markdown file or PDF format.

ADD the following statement to all your code files including the file named `ideas.pdf`.

This work is mine unless otherwise cited - Student Name

Part 04 - Do you want to take up more challenges?

Do you have more time and like to take up the additional challenge(s). Add an additional step to the `DCD1.java` and DCD2.java to measure the running time of the program for a variety of dataset sizes. After calculating the running time for different dataset sizes such as [100,200,300,400,500,600,700,800,900,1000] coin collections, prepare a table with the running time. In order to produce more stable results, the results may be averaged from multiple executions. Based on the table, produce some simple bar and/or line chart to show the performance comparison of both algorithms. If the performance comparison is not obvious, the dataset size may be increased. Although there are no points awarded for this part explicitly, these additional efforts would help a student to do well in other labs, skill tests, and/or exams.

The results may be shown using a tabular format and/or charts. The specification is spelled out not in detail so as to encourage students to be creative in their own way. An online tutorial such as the one provided below shows how to plot such a chart on google sheets.
`https://www.youtube.com/watch?v=sBohZdE0TIU`

There are several other online sources that illustrate how to draw different charts. It is an important skill to learn how to present experimental results from program implementation. The Data Analytics course in the CS department cover more topics in this direction. It is highly recommended to take that course if not done previously!

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 from the “diurnal-algorithm.tex” program.
2. Commented source code from the “diurnal-algorithm.pdf” program.
3. Commented source code from the “exp-tracker2.tex” program.
4. Commented source code from the “exp-tracker2.pdf” program.
5. Commented source code from the “DCD1.java” program.
6. Commented source code from the “DCD2.java” program.
7. A document containing the peer discussion points, named `pd-summary.pdf`.
8. A document containing the ideas, named `ideas.pdf`.
9. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is made before the deadline. 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 PDF files. In your summary file, please make sure to include your name and your partner’s name.

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.


Wish you
all the best
alamy stock photo