Lab 01 Specification – A Hand-on Exercise to practice Computational Constructs
50 points

Lab Goals

- Quick review of GitHub and git commands.
- Refresh your ability to write good code in Java.
- Think about how to solve a variety of programmatic challenges.
- Write two versions of a program by implementing different behaviors.

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 you have not done so already, please 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 will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- **GT chapter 01**

Assignment Details

Now that we have discussed some basics of computational constructs and implemented a few Java programs together in the last few lectures, it is your turn. In this lab, you will practice a variety of programs to retain the knowledge of conditional and iterative constructs. This includes modifying one or more code files to implement a series of functionalities.

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 test, exams, etc · · ·

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.

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 101** 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/YS4HLU6jjHhicBxVA`

Part 02 - A Simple Exercise on Basic Computational Constructs (15 points)

Students are required to complete this part only in the laboratory room, which is in **Alden 101**. 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.

Write a Java program that implements the `Currency Exchange Rate Converter` program using a series of requirements outlined below.

1. **ADD** the following statement to all your code files including the file named `Currency.java`.

   This work is mine unless otherwise cited - Student Name

2. For simplicity, let us make the `Currency Exchange Rate Converter` program work for four countries and the conversion table is shown below:

From Currency	To Currency	Amount
INR	USD	0.014
INR	CNY	0.097
INR	JPY	1.55
USD	INR	71.04
USD	CNY	6.87
USD	JPY	110.18
CNY	USD	0.15
CNY	INR	10.35
CNY	JPY	16.05
JPY	USD	0.0091
JPY	INR	0.64
JPY	CNY	0.062

Table 1: Exchange Rate Table

3. The starter code is provided inside the lab repository in a file named, `Currency.java`.
4. First, the program should greet the user with a welcome message. This is already implemented in the starter code file so as to get started and understand the other requirements outlined in this section.
5. After displaying the greeting message, the program should prompt the user to enter the from and to currency. To make this section easier to understand, this step is already implemented in the starter code as well.
6. Implement the exchange rate conversion rules based on the table provided in this section. The implementation should be constructed using if, else if, and else conditions.
7. For each of the conditions implemented in the previous step, implement a switch case block to formulate the rules for conversion from one currency to another.
8. Do some basic data validation procedure to correctly identify invalid inputs. For example, if user-provided input for both currencies is `1`, then an invalid input message should be displayed to the user. In general, it is acceptable to assume that the user-provided input for the currency specification is always a number between 1 to 4 (inclusive).
9. A screenshot displayed in the next page shows the welcome greeting message, the user prompt, and the output of the program to convert US Dollars to Indian Rupees.

```
Aravinds-MacBook-Pro-916:labs amohan$ javac Currency.java
Aravinds-MacBook-Pro-916:labs amohan$ java Currency
Welcome to Currency Exchange Rate calculator ....
-----------------------------------
The following options should be used to define a currency in our tool:
1 for Chinese Yuan
2 for Japanese Yen
3 for Indian Rupee
4 for United States Dollar
Tell us about the currency that you have by entering a number between 1 and 4?
4
Tell us about the currency that you want by entering a number between 1 and 4?
3
How much is the amount?
100
100 USD is 7104.000000000001 Indian Rupees
Aravinds-MacBook-Pro-916:labs amohan$
```

Part 03 - Pair Discussion (5 points)

Students are required to complete this part only in the laboratory room, which is in **Alden 101**. 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.

Pair up with at least one of your peers. Talk to your peer and discuss with them at least one idea of how to further enhance the `Currency.java` program. 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 the idea. The summary should also include the critic points from the peer. Name the summary file as **pd-summary.pdf**. Here **pd** refers to pair discussion. In order to pass the submission requirements, the file should be generated using a PDF format.

ADD the following statement to all your code files including the file named `pd-summary.java`.

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 **29th 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.

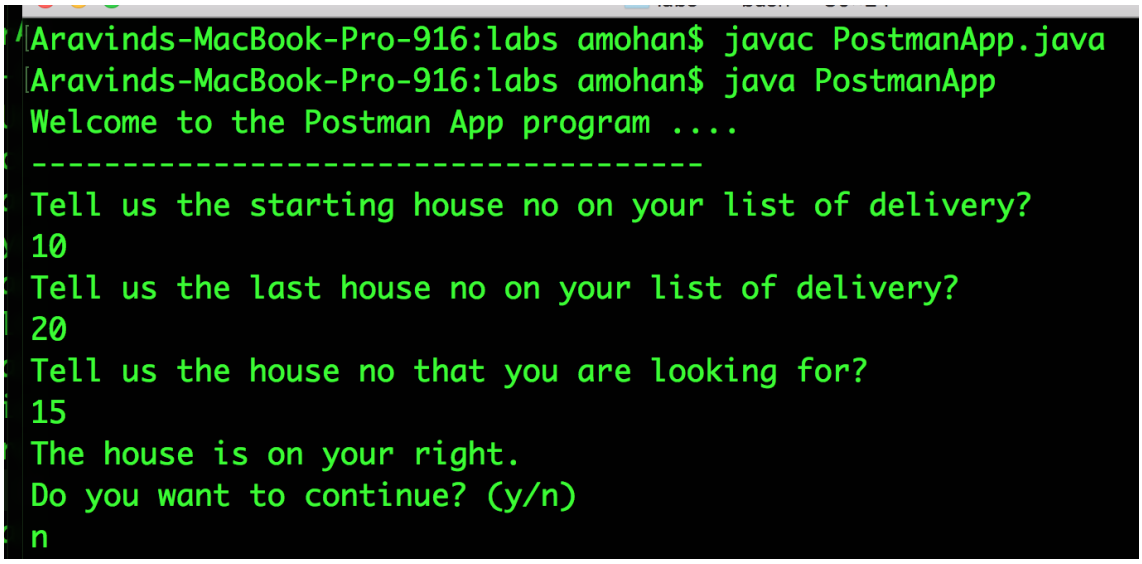
Part 01 - A Postman App (10 points)

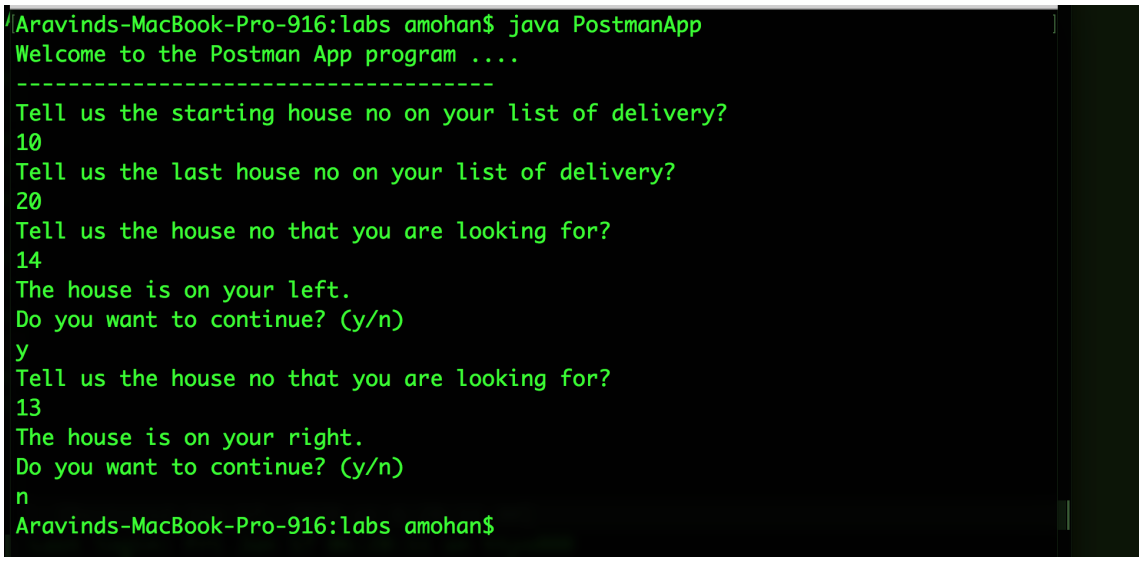
Write a Java program that implements a `Postman App` program using a series of requirements outlined below.

1. **ADD** the following statement to all your code files including the file named `PostmanApp.java`.

   This work is mine unless otherwise cited - Student Name

2. The starter code is provided inside the lab repository in a file named, `PostmanApp.java`.
3. First, the program should greet the user with a welcome message. This is already implemented in the starter code file so as to get started and understand the other requirements outlined in this section.
4. After displaying the greeting message, the program should prompt the user to enter the start and end house no in the postman's delivery list. To make this section easier to understand, this step is already implemented in the starter code as well.
5. Implement a repetitive block using a `while loop` to ask the postman the house no that he/she is currently looking for.
6. Based on a simple thumb rule, the odd house no's are located on the right side of the street and the even house no's are located on the left side of the street. The program should specify to the postman if the house is located on the left or right side of the street.
7. So how do we identify if a given house number is odd or even? As discussed in the class discussion last time, there exists an operator known as modulo (%) in Java which allows one to easily detect the odd and even number in the program.
8. The program should repeat the steps to guide the postman to locate a different house no. The program should exist if the postman specifies there is no more guidance needed.
9. The program should do some basic data validation procedure to correctly identify invalid inputs. For example, if the postman provided input for the house that he/she is looking for is not between the start and end house no's, then an invalid input message should be displayed in the console. In general, it is acceptable to assume that the postman provided input for the house that he/she is looking for to be a number between the start and the end house no's in the postman's delivery list (inclusive).
10. A screenshot displayed in next page shows the welcome greeting message, the user prompt, and a sample execution result of the program to detect the house location.


Aravinds-MacBook-Pro-916:labs amohan$ javac PostmanApp.java
Aravinds-MacBook-Pro-916:labs amohan$ java PostmanApp
Welcome to the Postman App program ....
Tell us the starting house no on your list of delivery?
10
Tell us the last house no on your list of delivery?
20
Tell us the house no that you are looking for?
15
The house is on your right.
Do you want to continue? (y/n)
n


Aravinds-MacBook-Pro-916:labs amohan$ java PostmanApp
Welcome to the Postman App program ....
Tell us the starting house no on your list of delivery?
10
Tell us the last house no on your list of delivery?
20
Tell us the house no that you are looking for?
14
The house is on your left.
Do you want to continue? (y/n)
y
Tell us the house no that you are looking for?
13
The house is on your right.
Do you want to continue? (y/n)
n
Aravinds-MacBook-Pro-916:labs amohan$

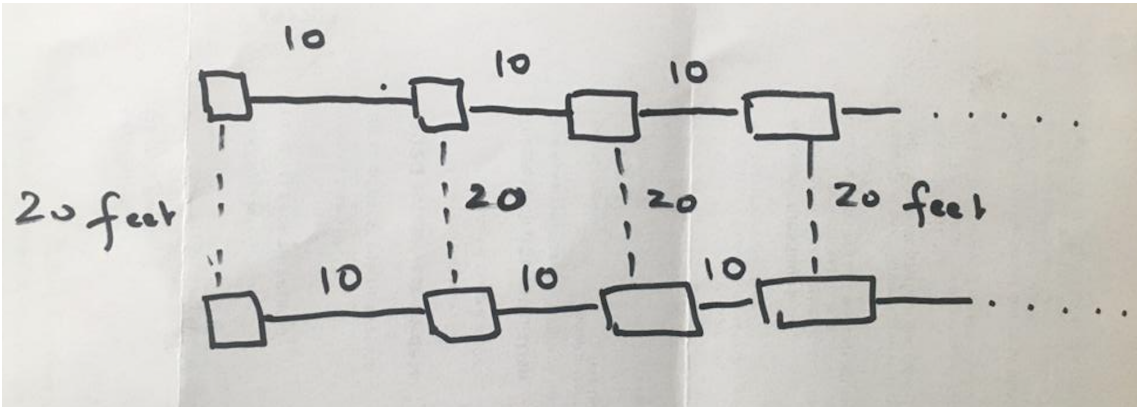
Part 02 - A Postman Location Report (10 points)

Write a Java program that implements a `Postman Location Report` program using a series of requirements outlined below.

1. **ADD** the following statement to all your code files including the file named `PostmanReport.java`.

   This work is mine unless otherwise cited - Student Name

2. The starter code is provided inside the lab repository in a file named, `PostmanReport.java`.
3. First, the program should greet the user with a welcome message. This is already implemented in the starter code file so as to get started and understand the other requirements outlined in this section.
4. After displaying the greeting message, the program should prompt the postman to enter the start and end house no in the postman's delivery list. In addition, the program should prompt the postman to specify the current house no that he/she is located at. To make this section easier to understand, this step is already implemented in the starter code as well.
5. Implement a repetitive block using a `while loop` to ask the postman the house no that he/she is currently located at.
6. As outlined in the previous section, based on a simple thumb rule, the odd house no's are located on the right side of the street and the even house no's are located on the left side of the street. The program should specify to the postman if the house is located on the left or right side of the street.
7. So how do we identify if a given house number is odd or even? As discussed in the class discussion last time, there exists an operator known as modulo (%) in Java which allows one to easily detect the odd and even number in the program.
8. The program should repeat the steps to generate a report for the postman to locate the houses in the postman's delivery list.
9. The program should make an assumption that the houses on the same side are 10 feet apart and the houses located opposite to each other are 20 feet apart. To clarify, a simple diagram of the arrangement of houses in the street is displayed on the next page:



10. The program should do some basic data validation procedure to correctly identify invalid inputs. For example, if the postman provided input for the house that he/she is currently located at is not between the start and end house no's, then an invalid input message should be displayed in the console. In general, it is acceptable to assume that the postman provided input for the house that he/she is currently located at to be a number between the start and the end house no's in the postman's delivery list.

11. The program should exist if the postman specifies there is no more guidance needed.

12. A screenshot in next page shows the welcome greeting message, the user prompt, and an example of generating a report of the houses and their distance and location.

```
Aravinds-MacBook-Pro-916:labs amohan$ javac PostmanReport.java
Aravinds-MacBook-Pro-916:labs amohan$ java PostmanReport
Welcome to the Postman Location Report program ....
-----------------------------------
Tell us the starting house no on your delivery list?
10
Tell us the last house no on your delivery list?
20
Tell us the house no that you are in now?
10
10      10      0 feet on same side...
10      11      20 feet on opposite side...
10      12      10 feet on same side...
10      13      30 feet on opposite side...
10      14      20 feet on same side...
10      15      40 feet on opposite side...
10      16      30 feet on same side...
10      17      50 feet on opposite side...
10      18      40 feet on same side...
10      19      60 feet on opposite side...
10      20      50 feet on same side...
Do you want to continue? (y/n)
y
```

```
Aravinds-MacBook-Pro-916:labs amohan$ java PostmanReport
Welcome to the Postman Location Report program ....
------------------------------------
Tell us the starting house no on your delivery list?
10
Tell us the last house no on your delivery list?
20
Tell us the house no that you are in now?
15
15      10      50 feet on opposite side...
15      11      20 feet on same side...
15      12      40 feet on opposite side...
15      13      10 feet on same side...
15      14      30 feet on opposite side...
15      15      0 feet on same side...
15      16      20 feet on opposite side...
15      17      10 feet on same side...
15      18      30 feet on opposite side...
15      19      20 feet on same side...
15      20      40 feet on opposite side...
Do you want to continue? (y/n)
n
```

Part 03 - To Think (5 points)

An important part to programming is to develop thinking skills. By now, we had implemented two interesting ideas connected to the life of a Postman. 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. 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 PDF format.

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

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 `PostmanReport.java` program file to output the result to a text file. If you would implement this step, it is important to provide the postman an option to chose either display the output on the console or store the output in a text file. 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.

A sample prompt would be:

```
Do you want the output to be stored in a text file? y/n
```

If the postman specifies "y" as input, then store the output in a text file, named `report.txt` inside the lab repository. Simply display the output on the console otherwise. The sample code and documentation provided below outlines the steps to be taken to write output in a text file.
`https://mkyong.com/java/how-to-write-to-file-in-java-bufferedwriter-example/`
`https://mkyong.com/java/java-how-to-append-text-to-a-file/`

Submission Details

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

1. Commented source code from the “Currency” program.
2. Commented source code from the “PostmanApp” program.
3. Commented source code from the “PostmanReport” program.
4. A document containing the peer discussion points, named `pd-summary.pdf`.
5. A document containing the ideas to enrich the functionality, named `ideas.pdf`.
6. 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 syllbus. 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 will be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your git repo, will lead you to receive 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 you need any clarification on your lab grade, talk to the Professor. The lab grade may be changed if deemed appropriate.