CS101- Algorithms and Programming I

Lab 08

Lab Objectives: arrays.

IMPORTANT:

- For all labs in CS 101, your solutions must conform to the CS101 style guidelines (rules!).
- You should not use break statements in your solution.
- Before implementing your programs, you should analyse the problem and plan and write the algorithm you will use to solve the problem.
- Do not use ArrayLists in your solution; the only data structure you should use are arrays.
- **Do not** sort the arrays.

Your friends are planning a party. Unfortunately, because you are very busy, you can only attend the party for one hour. However, you want to attend when the most friends will be there. You have asked your friends when they plan to attend the party and collected the information. Your program should define 3 parallel arrays:

- one array stores the names of your friends,
- one arrays store the priority of each friend from 1 (acquaintance) – 5 (bff)
- one stores the time they plan to arrive at the party, and
- one stores the time they plan to leave.

You may declare these arrays as static global variables inside the class and initialize them using the array initializer. You may define additional helper methods as necessary.

Assumptions:

- The party will be in the evening, so you can assume the arrival and departure times are between 1pm – 12am (everyone will leave by 12).
- The interval [arrive, depart) means that the friend will be at the party on and through the arrival hour, but will have left when the depart hour begins. For example, if you are available between 8 – 10, and your friends will be at the party from 6 – 8 or 10 – 12, you will not see those friends.

Below is the data used in the sample runs; you may use the same data or define your own:

Arrival times of friends: `7,5,8,7,9,7,8`
Departure times of same friends: `12,7,11,12,10,9,10`
Names of friends: `Ela,Eren,Alona,Jen,Mark,Mel,Ender`
Priorities: `1,1,5,2,2,4,3`

1. Write a method, `friendsPriority()`: you should pass to the method the hours which you are available (for example, between 7 – 10) that returns an array where each element contains the average priority of all friends who will be in attendance for each of the given hours. For example, if you decide you can attend between 7 – 10, the method will return an array with 3 elements, [2 .33, 3.0, 2.6], indicating the average priority of all friends in attendance at 7pm, 8pm and 9pm.

2. Write a method, `bestTimeToAttend()`, that returns the best time to attend the party. You should pass to the method the hours which you are available (for example, between 6 – 10). However, you had a fight with one of your friends, and you do not want to attend the party when they are there, so you should also pass the name of the friend who you wish to avoid. The method should return the hour that you should attend with the highest priority (closest) friends, where the friend to avoid is not in attendance. The method should return -1 if no suitable hour is found.

   Using this method, display the best time to attend to see the most friends and avoid the given friend.

   Sample Run:

```
Enter friend you wish to avoid: Mel
Enter time interval you are available: 7 10
Best time to attend to avoid Mel and to see the closest friends is 9
```