Homework 6

Due by **12:30pm, Friday, November 4, 2016**.

Make sure you follow all the homework policies (http://www-student.cse.buffalo.edu/~atri/cse331/fall16/policies/hw-policy.html).

All submissions should be done via Autolab (http://www-student.cse.buffalo.edu/~atri/cse331/fall16/autolab.html).

1. Lest you think I have anything against liberals, let me assure you that my brother-in-law has “termed" me and my wife as being liberal elites.

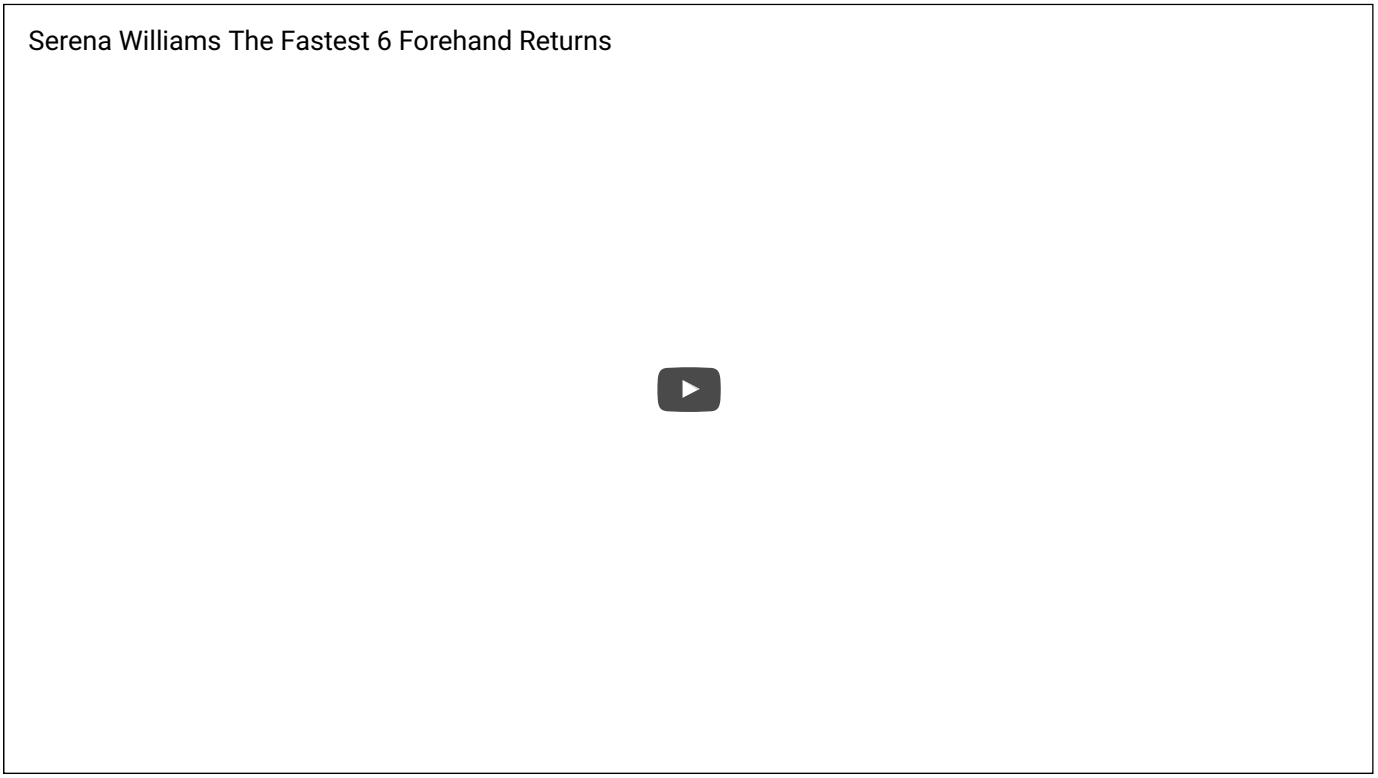
Question 1 (Programming Assignment) [40 points]

</> Note

This assignment can be solved in either Java, Python or C++ (you should pick the language you are most comfortable with). Please make sure to look at the supporting documentation and files for the language of your choosing.

The Problem

The `TennisSuperVillian` is hell-bent on destroying earth. Humans have one last hope: the great tennis player Serena Williams (https://en.wikipedia.org/wiki/Serena_Williams):



Serena has two choices to neutralize `TennisSuperVillian`. The first choice is to nuke `TennisSuperVillian`'s hideout. The second choice is to defeat the `TennisSuperVillian` in his own game. Serena has to beat `TennisSuperVillian` in n special rallies. The i^{th} rally ($0 \leq i \leq n-1$) needs a given amount of time t_i and it has to be finished by time f_i. Each of these rallies are complex and (even) Serena can only play one rally at

a time and each rally once it has been started has to be completed without interruption (otherwise `TennisSuperVillian` wins and the earth gets destroyed). `TennisSuperVillian` is cunning and has made sure that the only way to defeat him is to finish all rallies i by time f_i . You should also assume that Serena cannot start any of the rallies before time 0. (Also assume that all the t_i and f_i are positive integers.)

Serena also has intelligence that `TennisSuperVillian` has planned an imminent attack soon and she does not have time to try out both choices.

Obviously Serena wants to avoid the first choice because of the collateral damage. However, as mentioned earlier time is short and Serena needs to decide if she can go with the second choice. During her training, she skipped the class on greedy algorithms. Your task is to design an algorithm for Serena that will help her make her decision in time $O(n \log(n))$.

Input

- The first line of the input file will have the number n, indicating the number of rallies
- The next n lines will have the durations (t_i) and deadlines (f_i) for the n rallies (separated by spaces)
- The index of the line under consideration is the ID of that rally. For instance line 0 of the input file (not including the line displaying n) has the duration (t_0) and deadline (f_0) of rally 0

```
n                <- Number of rallies Serena has to play
t0 f0            <- Duration t0 and deadline f0 of rally 0
t1 f1            <- Duration t1 and deadline f1 of rally 1
t2 f2            <- Duration t2 and deadline f2 of rally 2
.
.
.
tn-1 fn-1                <- Duration tn-1 and deadline fn-1 of rally n-1
```

For example:

```
6              <- Serena needs to play 6 rallies
5 16           <- Rally 0 has a duration of 5 and needs to finish by 16
7 18           <- Rally 1 has a duration of 7 and needs to finish by 18
10 28          <- Rally 2 has a duration of 10 and needs to finish by 28
12 45          <- Rally 3 has a duration of 12 and needs to finish by 45
4 11           <- Rally 4 has a duration of 4 and needs to finish by 11
3 29           <- Rally 5 has a duration of 3 and needs to finish by 29
```

Output

If you decide that Serena should play out all the rallies, you should output a list of pairs (rally ID, rally start time).

If you decide that the only option Serena has is to nuke `TennisSuperVillian` 's hideout, output an empty list.

```
[(0, s0), (1, s1), .... , (n, sn)]        <- rally 0 starts at time s0
                                             rally 1 starts at time s1
                                             .
                                             .
                                             .
                                             rally n starts at time sn
```

For example, using the example input from above:

```
[(4, 0), (0, 4), (1, 9), (2, 16), (5, 26), (3, 29)]            <- rally 4 starts at time 0
                                                                  rally 0 starts at time 4
                                                                  rally 1 starts at time 9
                                                                  rally 2 starts at time 16
                                                                  rally 5 starts at time 26
                                                                  rally 3 starts at time 29
```

</> Note

If your algorithm decides that Serena should play out all the rallies, your output schedule might not match the corresponding output file. That is fine. As long as all the rallies in your schedule finish by their respective deadlines, the grader will give you full points.

! Note

Both the input and output parsers in each of the three languages are already written for you.
Note that you have to work with the input data structures provided (which will come pre-loaded with the data from an input file).

! Addition is the only change you should make

Irrespective of what language you use, you will have to submit just one file. That file will come pre-populated with some stuff in it. You should **not** change any of those things because if you do you might break what the grader expects and end up with a zero on the question. You should of course add stuff to it (including helper functions and data structures as you see fit).

Java | Python | C++

Download Java Skeleton Code (HW6Java.zip)

Directory Structure

```
├── Driver.java
├── HW6_StudentSolution.java
├── testcases/
│   ├── input1.txt
│   ├── input2.txt
│   ├── input5.txt
│   ├── output1.txt
│   ├── output2.txt
│   └── output5.txt
```

You are given two coding files: `Driver.java` and `HW6_StudentStudent.java`. `Driver.java` takes the input file, parses it and creates an instance of the class and prints the result to the command line. You only need to update the `HW6_StudentSolution.java` file. You may write your own helper methods and data structures in it.

The testcases folder has 3 input files and their corresponding output files for your reference. We will use these three input files (and seven others) in our autograding. The output files provided are not the ONLY solutions. They only represent ONE possible way Serena could schedule her rallies.

Method you need to write:

```
 5.
10.
```

```java
/**
 * You should fill this in on your own. Rallies is list of tuples, in the form (
 * Your output should also be a list of tuples, of the form (rally id, start time
 * schedule exists, you should return an empty list.
 * @return
 */
public ArrayList getSchedule() {
    ArrayList schedule = new ArrayList<>();

    return schedule;
}
```

The `HW6_Student` class has 1 instance variables:

- `private ArrayList<int[]> rallies;` which is a list of tuples representing the input from the file. The pairs are of the form (rally duration, rally deadline).

The output of the code has the same data structure. However, this time, the tuples should be of the form (rally id, start time of rally).

Compiling and executing from command line:

Assuming you're in the same directory level as `Driver.java`. Run `javac Driver.java` to compile.

To execute your code on `input1.txt`, run `java Driver testcases/input1.txt`. The output array will be printed to `stdout`.

Submission

You only need to submit `HW6_StudentSolution.java` to Autolab.

Grading Guidelines

We will follow the usual grading guidelines for programming questions (../../policies/hw-policy.html#grading).

Question 2 (Buffalo to Seattle) [45 points]

The Problem

Ms. LiberalElite likes driving her Prius, which on one full tank of gas runs for 350 miles. [1] She is planning to drive from Buffalo to Seattle and has fixed up her driving route so that she can get to Seattle as fast as possible. She also has a map with the locations of all the n gas stations along the way. You can assume that the route is a straight line and the gas stations are points on the line. Ms. LiberalElite missed the class on greedy algorithms, so in this problem you will design an efficient algorithm for her, which she can use to figure out which gas stations she should stop at so that she stops at the minimum number of gas stations. Of course your algorithm should give a feasible set of gas stations, i.e. it should never be the case that is she is stranded between two gas stations without any gas. (You may assume that Ms. LiberalElite fills up the tank of her Prius whenever she stops at a gas station and that no two gas stations are more than 349 miles apart. Also you can assume that she starts off from Buffalo with a full tank.)

Prove the correctness of your algorithm and analyze its running time. (For the latter, you will get more credit the smaller the running time of your algorithm.)

Hint

Think of a greedy way to decide on which gas stations to stop. It might help to forget about the scheduling algorithms we have seen in class and just think of a greedy algorithm from "scratch." Then try to analyze the algorithm using similar arguments to one we have seen in class.

Submission

You need to submit **one PDF** file to Autolab. We recommend that you typeset your solution but we will accept scans of handwritten solution-- you have to make sure that the scan is legible. Also make sure that you preview your upload on Autolab to make sure it was uploaded correctly.

Grading Guidelines

We will follow the usual grading guidelines for programming questions (../../policies/hw-policy.html#grading). Here is a high level grading rubric specific to this problem:

1. `Algorithm idea: 10` points.
2. `Algorithm details: 10` points.
3. `Proof of correctness idea: 10` points for arguing correctness of the algorithm.
4. `Proof details: 10` points.
5. `Runtime analysis: 5` points.

! Note

If you do not have labeled and separated out proof idea, algorithm idea, proof details, algorithm details and runtime analysis you will get a zero(0) irrespective of the technical correctness of your solution..

Question 3 (Chap. 4, Ex. 6) [15 points]

The Problem

Exercise 6 in Chapter 4 of the textbook.

Note

As has been mentioned earlier, in many real life problems, not all parameters are equally important. Also sometimes it might make sense to "combine" two parameters into one. Keep these in mind when tackling this problem.

Hint

In the solution that I have in mind, the analysis of the algorithm's correctness follows the exchange argument. Of course that does not mean it cannot be proven in another way.

Submission

You need to submit **one PDF** file to Autolab. We recommend that you typeset your solution but we will accept scans of handwritten solution-- you have to make sure that the scan is legible. Also make sure that you preview your upload on Autolab to make sure it was uploaded correctly.

Grading Guidelines

We will follow the usual grading guidelines for programming questions (../../policies/hw-policy.html#grading). Here is a high level grading rubric specific to this problem:

1. `Algorithm idea : 3` points.
2. `Algorithm details : 3` points.
3. `Proof of correctness idea : 3` points for arguing correctness of the algorithm.
4. `Proof details : 3` points.
5. `Runtime analysis : 3` points.

> **! Note**
>
> **If you do not have labeled and separated out proof idea, algorithm idea, proof details, algorithm details and runtime analysis you will get a zero(0) irrespective of the technical correctness of your solution.**.