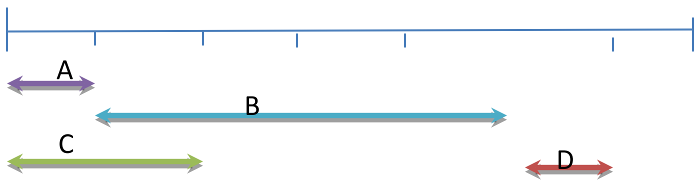
Week 3 Assignment: Task Scheduling (30 pts)

Problem Description

Suppose that you are scheduling a room. You are given a group of activities each of which has a start and stop time. Two activities are compatible if they do not overlap (one activity finishes before another one starts). For example, in the following activities, activity A is compatible with activities B and D but not activity C:

Activity	**Start Time**	**Stop time**
A	**1**	**2**
B	**2**	**5**
C	**1**	**3**
D	**5**	**6**



The room has a start time and an end time in which it is available.

Your goal is to write a recursive method to schedule compatible activities that result in the maximum usage of the room. The usage of the room is defined as the total duration of the scheduled activities, that is, the sum of (stop time – start time) for all the activities scheduled to run in the room.

For example suppose that the start time and end time in which the room is available is [1,7] for the above table. Hence, the possible schedules are:

1. Activities A, B,D: with room usage = (2-1)+(5-2) +(6-5) = 5
2. Activities C, D: with room usage = (3-1)+(6-5) = 2
3. Activities A,D: with room usage (5-2)+(6-5) =4
4. Activities A, B: with room usage (2-1)+(5-2)= 4
5. Activities B, D: with room usage (5-2)+(6-5) = 4

Therefore, the set of activities {A,B,D} gives the optimal schedule with the maximum room usage.

What you need to do:

1- **Create a class Activitiy.java** with three data fields : activityName, startTime, and stopTime . Create accessor (get) and mutator (set) methods for each data field.

2- **Create a class Scheduling.java** with the following method in it:

```
    public Activity[] getOptimalSchedule( int roomStartTime, int roomEndtime,
Activity[] activities)
```

This method gets the availability span of the room as well as a set of activities to choose from and returns an optimal schedule i.e., a selected set of non-overlapping activities that can be scheduled to run in the room and gives the maximum possible usage of the room. Note, depending on your input, the correct answer may not be unique (that is, there might be several valid schedules with the same total room usage)

Note: it is important that your getOptimalSchedule method has the signature above for ease of testing and grading. Also you can use any additional utility methods in your classes as long as you provide comments explaining what each utility method does.

What you need to turn in:

You need to submit your two java files : **Activity.java** and **Scheduling.java**

Attention: To get a full credit, your algorithm must be recursive.

Grading Rubric:

Criteria	Points
The Activity.java class is correctly written with correct data fields and accessor and mutator methods	5 pts
getOptimalSchedule correctly returns an optimal schedule with maximum room usage	10 pts
getOptimalSchedule has a correct base case and terminates without error	5pt
getOptimalSchedule has correct recursive calls	10 pts

Hint:

Consider two recursive cases 1- The first activity is not included in the final optimal schedule, and 2- the first activity is included in the final optimal schedule. Solve the problem recursively for each of these two cases to get the optimal schedule for each case and compare their room usages. The final answer will be the optimal schedule of the case with the higher room usage. Please refer to the practice problem 4 in module 3 lectures for an example of a problem with a similar recursive logic.