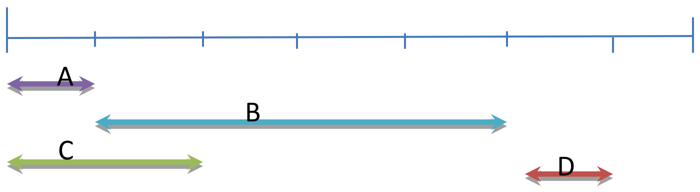
Week 3 Assignment (Option 2) : Task Scheduling

22 points

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 ArrayList<Activity> getOptimalSchedule( int roomStartTime, int
roomEndtime, ArrayList<Activity> activities)
```

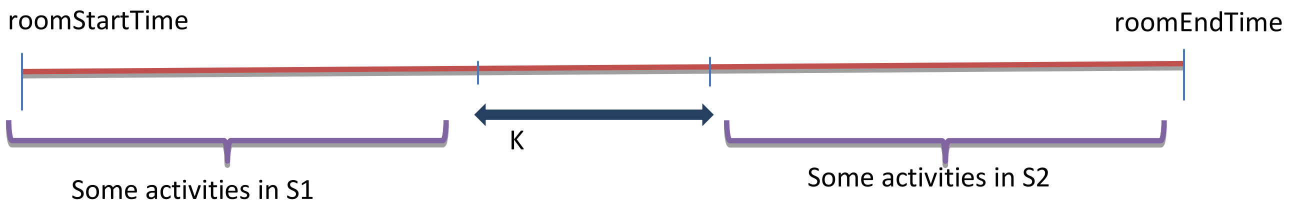
This method gets the availability span of the room as well as a list of activities to choose from and returns an optimal schedule i.e., a selected set of non-overlapping activities that can be scheduled to run in the room and gives the maximum possible usage of the room.

Note: 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.

ArrayList is a dynamic array that can grow dynamically as you add elements to it. You must already be familiar with the ArrayList class in the standard java library from your JAVA II class. If not, you can take a look at this tutorial (http://www.tutorialspoint.com/java/java_arraylist_class.htm) to see how you can create and add/remove elements to/from an ArrayList.

Hint:

To solve this problem recursively, suppose that we know that the solution contains activity k. Now we are left with two sets of possible activities to choose from : S1={the set of all activities that finish before K starts } and S2={the set of all activities that start after K is finished}:



We can solve the problem recursively twice, once for S1 and once for S2. This will give us an optimal schedule for S1 and an optimal schedule for S2. So assuming that the solution contains activity K, the optimal schedule will be: {optimal schedule for S1} $\cup$ {K} } $\cup${optimal schedule for S2}

The total room usage then will be the room usage for the optimal schedule for S1 + length of K + the room usage for the optimal schedule for S2.

Since we don't know for sure which activity K is included in solution, we can try each activity in the list of activities for K and return the schedule that gives us the maximum room usage.

What you need to turn in:

Write a main method to test your getOptimalSchedule method. Once you are sure that your method works correctly, submit your two java files : **Activity.java** and **Scheduling.java**

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