CS360 – Spring 2015
Project 4: Activity Scheduling

Using either Java (6/7) or C++ 1998 specification, write a program that solves the following problem.

Input:
Get input from standard in. The first record will contain two values separated by commas. The first value will be the end of the schedule. The second value will be number of activities. You can assume that the value for the activities correctly gives the number of activities available. You can assume that the value given for the end of the schedule is at least as large as the last finish time. Do not make any assumptions about the size of the number other than it can fit in an integer type. Both values will be integers.

Subsequent lines will provide activities information. Each activity will have a name, a start time and an end time. Start time will always be before end time. Start and finish times will be integers. All fields are separated by commas. Create your own data based on specs. Below is just a simple sample. You can assume the activity names are unique (only to make verifying output easier). There is no sorting guarantee.

Input sample

```
100,13
CS100,9,15
CS101,12,23
CS102,22,33
CS103,32,43
CS104,42,53
CS105,52,63
CS106,62,73
CS107,72,83
CS108,92,93
CS109,92,98
CS110,25,50
CS111,55,75
CS112,2,3
```

Functionality
Calculate an optimal schedule in terms of activity number using a greedy algorithm. Print the schedule out in finish order as shown in output. Print the schedule in start time order. Only arrays can be used. Any other data structures must be custom.

Output sample

```
Schedule
CS112 from 2 to 3
CS100 from 9 to 15
CS102 from 22 to 33
CS104 from 42 to 53
```

Failure to follow directions will result in point deductions or a zero. There are 75 students in this class. It is unreasonable to expect that any exceptions to the procedure will be made.

Late assignments will not be accepted unless you have a doctor's note covering the entire period from Mar 3 – Mar 10th. Java programs should not require any machine specific code. The C++ programs will be tested on bama.ua.edu. You can gain access to this system for testing. Develop C++ on other platforms at your own risk. Programs must only include constructs made available by the language specifications (Java 6 and C++11/Posix- https://en.wikipedia.org/wiki/C%2B%2B98) and no other extensions. You also must comply with any prohibitions on data structures in the program description.

Blackboard submission by Mar 11 @11:59pm:
• Program code in ASCII file with .java or .cpp extension. Pdfs of code will not be graded. Class files or executables will not be graded
• Test results in pdf format. You should generate test data such that it shows that your program works.
• If your solution utilizes more than one source file (and there is good reason why it would), provide a Unix batch (bash or csh), make or ant file to compile and run the test cases. File should run without any machine specific requirements other than make, bash/csh and ant.
• Any relevant notes in text file README

In class printout submission Mar 12 @ 9am:
• Program printout that is easy to read (line wrapping etc).
• Short version of test results (save a tree...)
• The source code (.java or .cpp files) should also be turned in via Blackboard (not emailed to me or the TA). I do not use a Windows computer. Test result submissions of any type other than pdf will not be graded.
•***No packages or directories used (default package running current directory)***
• If you forget to turn in your test results, your program may be graded as if it does not work.
• The source code and test results should be printed and brought to class Mar 11th at 9am to get credit. Any assignments that are not submitted via printout on Mart 11th will receive a 50% deduction.
• I will do some initial automated testing based on checking the produced output. Adhere to the output format closely.
• This is an individual assignment. The program must represent your own work. You can discuss high-level concepts. Do not show your code to anyone. I reserve the right to ask you about your program to discern if you indeed wrote the code. If you cannot explain your code and choices verbally, you may be turned in for academic misconduct. All submissions will be analyzed to identify possible cases of cheating. Any cases of suspected collaboration will be referred to the College of Engineering

Dean. A zero or low grade is always better than having an academic misconduct on your academic record.

Here is an overview of evaluation criteria. It is subject to change within 10% at my discretion. The top submissions that contain particularly elegant and clear code may earn extra credit.

********* Programs that use approaches that are asymptotically slower than the optimal, contain extra loops or any code that hurts efficiency will receive a zero regardless of functionality. This assignment (albeit short) should showcase all that you have learned.**

50% - Functionality
• Program contains the correct code to produce the output.
• Program meets the requirements (correct calculations and categorizations, data structure, etc)
• Grabs input from STDIN
• No artificial limits on the schedule, activity number or timeline
• Output from running the program is included that shows that the program works
• Output in correct format
• Only uses approved data structures
• Appropriate sort used

40% - Design
• Program uses appropriate language constructs
• Overall design should be reasonably consistent with OO design principles
• Program displays a reasonably fast method for producing the output
• Program exhibits defensible design choices in algorithms and data structures (if you add any)

10% - Style (Checked vigorously)
• Program must use appropriate and consistent style for naming of elements (C style or c++ style)
• Program must include reasonable whitespace and appropriate indentation
• Program must include comments, especially in areas where you need to support your choices or where the purpose of the code is unclear.

** Clarifications on the assignment will be posted to blackboard. You will be responsible for consulting blackboard frequently and remaining aware of clarifications.