Algorithms CS 336 **Problem Set #2**
Date: Friday, January 17, 2020 **Due:** Thursday, January 23, 2020

Submit your solution on Canvas.
Do not discuss these problems with other students. You should solve these problems on your own.

Problem 1. You are asked to design a computer program for a student art club. The members of the club want to attend all temporary exhibitions in the Metropolitan Museum of Art in New York City in 2020. To do so, students will travel to NYC several times. Every time they will stay in NYC for one day. On that day, students can attend **all** temporary exhibitions that are on display. The schedule of temporary exhibitions is known in advance (for each exhibition, you know the first and last days of the exhibition). Your goal is to schedule the minimum number of trips to NYC so that students can attend all temporary exhibitions.

I. Design and describe a greedy algorithm for this problem.

II. Analyze its running time.

III. Prove that the algorithm is correct.

Problem 2. Consider a variant of the Interval Scheduling Problem discussed in class. Suppose that we are given two arrays of jobs. The first array (highPriorityJobs) contains high priority jobs and the second array (lowPriorityJobs) contains low priority jobs. Your goal is to design an algorithm that schedules **all** high priority jobs and the maximum possible number of low priority jobs. Please, implement the following function:

int FindOptimalSchedule(std::vector<Job> highPriorityJobs, std::vector<Job> lowPriorityJobs)

The function should return the number of scheduled **low priority jobs**.

Instructions for the programming assignment. Download files

- `student_code_2.h` – this file should contain your solution.
- `problem_solver_2.cpp` – this is the main file in the project (don't edit this file!).
- `test_framework.h` – this is a library responsible for reading and writing data files (don't edit this file!)
- `problem_set_2.in` – this file contains test problems for your algorithm (don't edit this file!)

Place all files in a new folder/directory. Write your code in function `FindOptimalSchedule`. Also, write your name in the function `GetStudentName`. Both functions are located in file `student_code_2.h`. Compile and run your code. To compile your code do the following.

- If you use GNU C++ compiler, type
  ```
  g++ -std=c++11 problem_solver_2.cpp -o problem_solver_2
  ```
- If you use CLang compiler, type
  ```
  clang++ -std=c++11 problem_solver_2.cpp -o problem_solver_2
  ```
- If you use Microsoft Visual C++ compiler, start `Developer Command Prompt` and type
  ```
  cl /EHsc problem_solver_2.cpp
  ```

Your compiler should be compatible with `C++11`. If you work in the Wilkinson Lab[1], you need to start developer tools first: Type

- `scl enable devtoolset-4 bash`

Once you compile your code, start your program. Type `./problem_solver_2` on Unix or Mac and `problem_solver_2.exe` on Windows. Make sure that the executable is located in the same folder as file `problem_set_2.in`. Your program will generate `solution_2.dat` that contains solutions to the problems from file `problem_set_2.in`. If your code works correctly, you will get the following message:

- Problem set 2. Your algorithm solved all test problems correctly. Congratulations!
- Don't forget to submit your source code and file `solution_2.dat` via Canvas.

If your code makes a mistake, you may get a message like this:

- `Problem set 2.  Mistake in problem #15.  Correct answer:  4.  Your answer:  12.`

Finally, when your code is ready, submit files `student_code_2.h` and `solution_2.dat` via Canvas. Make sure that you are submitting the latest versions.

Remark: If you want to debug your code, please, type `./problem_solver_2 15` on Unix or Mac and `problem_solver_2.exe 15` on Windows. This command will call your function only on one problem – the problem #15 and thus let you debug your code on the problem where your program erred. Note that this command will not generate or update `solution_2.dat`. So before submitting your solution, you need to run your program without any command line arguments.

[1] You can ssh to *finagle.wot.eecs.northwestern.edu*