**Algorithms EECS 336** **Problem Set #5**
**Date:** Wednesday, February 13, 2019 **Due:** Tuesday, February 19, 2019

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

**Problem 1.** In this exercise, your goal is to design an algorithm for a cryptocurrency startup. The startup has a supercomputer that can mine two types of coins: NorthCoins and WestCoins. The profit from mining these coins drastically varies over time, and your goal is to determine which of the two types of coins the startup needs to mine at what time. The problem is that the supercomputer cannot immediately switch from mining NorthCoins to mining WestCoins or vice versa, because each type of coins requires its own software, and it takes 1 unit of time to load it. So, for example, if we mine NorthCoins at the time interval $[t, t+1]$, we can start mining WestCoins only at time $t+2$. In this particular case, the supercomputer will be loading software for WestCoins and not mining any coins at time interval $[t+1, t+2]$ .

Your algorithm is given two arrays of positive numbers $N_1, \ldots, N_T$ and $W_1, \ldots, W_T$.

- $N_t$ is the profit the startup gets from mining NorthCoins at interval $[t, t+1]$; and
- $W_t$ is the profit the startup gets from mining WestCoins at interval $[t, t+1]$.

Your algorithm needs to find the maximum profit the startup can get from mining NorthCoin and WestCoin at time interval $[1, T]$.

**Example.** *Consider the following example:* $N = \{10, 10, 10, 10, 20\}$ *and* $W = \{2, 15, 1, 7, 100\}$.

| | *1* | *2* | *3* | *4* | *5* |
|---|---|---|---|---|---|
| *N* | *10* | *10* | *10* | *10* | *20* |
| *W* | *2* | *15* | *1* | *7* | *100* |

*Then, the optimal solution to the problem is to mine Northcoins at time intervals* $[1, 2]$, $[2, 3]$, $[3, 4]$, *load new software at time interval* $[4, 5]$, *and mine WestCoins at time interval* $[5, 6]$. *The profit from this solution is* $10 + 10 + 10 + 100 = 130$.

1. Define a subproblem or several subproblems.
2. Describe the base cases.
3. Write a recurrence relation.
4. Prove that your recurrence relation is correct.
5. Give an algorithm for finding a solution to your subproblem.
6. Give an algorithm for finding the optimal solution to the original problem.
7. Analyze the running time of your algorithm.

**Hint:** You may define not one but three subproblems `ProfitN`$[t]$, `ProfitW`$[t]$, and `ProfitLoad`$[t]$. In every feasible solution to `ProfitN`$[t]$, the supercomputer mines NorthCoins at time $t$; in every feasible solution to `ProfitW`$[t]$, the supercomputer mines WestCoins at time $t$; in every feasible solution to `ProfitLoad`$[t]$, the supercomputer loads new software to mine NorthCoins or WestCoins.

**Remark:** You do not need to explain how backtracking works.

**Problem 2.** We ask you to implement your algorithms for Problem 1 in function

- `int FindMaxProfit (std::vector<int> north, std::vector<int> west)`

Arrays *north* and *west* contain profits the startup can get from mining NorthCoin and WestCoin (respectively). Note that arrays *north* and *west* are indexed from 0 to $T-1$, here $T$ is the size of the arrays. `FindMaxProfit` should return the maximum profit startup can earn at time interval $[0, T]$.

**Instructions for the programming assignment.** Download files:

- `student_code_5.h` – this file should contain your solution.
- `problem_solver_5.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_5.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 `FindMaxProfit`. Also, write your name in the function `GetStudentName`. Both functions are located in file `student_code_5.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_5.cpp -o problem_solver_5`
- If you use CLang compiler, type
  `clang++ -std=c++11 problem_solver_5.cpp -o problem_solver_5`
- If you use Microsoft Visual C++ compiler, start `Developer Command Prompt` and type
  `cl /EHsc problem_solver_5.cpp`

Your compiler should be compatible with `C++11`. If you work in TLab, you need to start developer tools first: Type

- `scl enable devtoolset-4 bash`

Once you compile your code, start your program. Type `./problem_solver_5` on Unix or Mac and `problem_solver_5.exe` on Windows. Make sure that the executable is located in the same folder as file `problem_set_5.in`. Your program will generate `solution_5.dat` that contains solutions to the problems from file `problem_set_5.in`. If your code works correctly, you will get the following message:

- Problem set 5. Your algorithm solved all test problems correctly. Congratulations!
- Don't forget to submit your source code and file `solution_5.dat` via Canvas.

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

- `Problem set 5. Mistake in problem #15. Correct answer: 4. Your answer: 12.`

Finally, when your code is ready, submit files `student_code_5.h` and `solution_5.dat` via Canvas. Make sure that you are submitting the latest versions.

**Remark:** If you want to debug your code, please, type `./problem_solver_5 15` on Unix or Mac and `problem_solver_5.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_5.dat`. So before submitting your solution, you need to run your program without any command line arguments.