CS 336: Algorithms **Problem Set 4**
Date: Saturday, October 30, 2021 **Due:** Thursday, November 4, 2021

Submit your solution on Canvas.

Problem 1. You are asked to design an algorithm for the game $1 * 3 * 5$. In this game, the player moves the piece on a rectangular board consisting of n squares (see below). There is a number written in every square of the board. We denote the number in square i by r_i. In the beginning of the game, the piece is located on square 1. Then, the player makes moves by advancing his or her piece by 1, 3, or 5 squares to the right. The player gets a reward for every move he or she makes. The amount of the reward depends on (1) the number written in the square where the piece stops and (2) the distance the piece travels during the move. Specifically, if the player moves the piece by one square, and the piece lands on square i, he or she receives r_i dollars; if the player moves the piece by three squares, he or she receives $3r_i$ dollars; if the player moves the piece by five squares, he or she receives $5r_i$ dollars. The game ends when the player cannot move the piece. Design a dynamic programming algorithm that finds the maximum reward the player can get. The input of the algorithm is the array $r_1, \ldots, r_n$.

I. Define a subproblem for your dynamic program (DP). Make sure that your definition is concise and unambiguous.

II. Describe the base case for your DP.

III. Write the recurrence relation.

IV. Explain why the recurrence relation is correct.

V. Show how to solve the original problem (i.e., the problem of finding the maximum possible reward) using the solution to the DP.

Problem 2. John wants to attend n football games this season. In order to so, he needs to get tickets for each game. He can buy two types of tickets: single game tickets and tickets for five *consecutive* games.

- A single game ticket is valid for one game. The ticket for game i costs p_i coins. John can use the discount code "CS 336" to get a 10% discount for every single game ticket (if p_i is not divisible by 10, the discount is rounded down to the closest integer). So, John's price for a single day ticket for day i is $p_i - \lfloor p_i/10 \rfloor$.
- A five game ticket is valid for five *consecutive* days. There are no discounts for these tickets. So, the price of the ticket for games i, $i+1$, $i+2$, $i+3$, and $i+4$ is $p_i + p_{i+1} + p_{i+2} + p_{i+3} + p_{i+4}$ coins.

Additionally, the ticket office charges John a fee of f coins for every ticket he buys (the fee is the same for single game and five game tickets).

Design an algorithm that finds the minimum cost of tickets for games 1 through n. The cost should include all fees and discounts. Note that John must buy *exactly one* ticket for each game.

Write the following function

```
MinCost(const std::vector<int>& prices, int fee)
```

The parameters of this function are `prices` – the prices of the tickets (`prices[i]` is the price for day i); `fee` –the ticket office fee charged for every ticket.

Collaboration policy for Problems 1 and 2: Please, solve these problems on your own. Do not collaborate with other students.

Instructions for the programming assignment.

Download files

- `student_code_4.h` – this file should contain your solution.
- `problem_solver_4.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!)
- `small_problem_set_4.in` and `large_problem_set_4.in` – these files contain test problems for your algorithm (don't edit these files!)

Place all files in a new folder/directory. Write your code in function `MinCost`. Also, write your name in the function `GetStudentName`. Both functions are located in file `student_code_4.h`. Compile and run your code. To compile your code do the following.

- If you use Clang compiler, type
  `clang++ -std=c++17 -pedantic-errors problem_solver_4.cpp -O2 -o problem_solver_4`
- If you use GNU C++ compiler, type
  `g++ -std=c++17 -pedantic-errors problem_solver_4.cpp -O2 -o problem_solver_4`
- If you use Microsoft Visual C++ compiler, start `Developer Command Prompt` and type
  `cl /EHsc problem_solver_4.cpp`

Your compiler should be compatible with `C++17`. If you work in the Wilkinson Lab, you need to start developer tools first: Type

- `scl enable devtoolset-4 bash`

Once you compile your code, start your program. Type `./problem_solver_4 small` to run your code on simple problems and `./problem_solver_4 large` to run your code on hard problems. On Windows, type `problem_solver_4.exe small` and `problem_solver_4.exe large`, respectively. Make sure that executable is located in the same folder as files `small_problem_set_4.in` and `large_problem_set_4.in`. If your code works correctly, you will get the following message:

> Problem set 4. Your algorithm solved all test problems correctly. Congratulations! `solution_4.dat` via Canvas.

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

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

Please, test your code with the both problem sets (small and large). When your code is ready, submit file `student_code_4.h` on Canvas. Make sure that you are submitting the latest versions!