CS 336: Algorithms **Problem Set 5**
Date: Saturday, November 6, 2021 **Due:** Thursday, November 11, 2021

Submit your solution on Canvas.

Problem 1. Consider a variant of the Weighted Interval Scheduling Problem discussed in class. Suppose we are given a set of jobs $1, \ldots, n$. Each job j has a start time s_j, processing time p_j, and weight w_j. In this variant of the problem, each job can be scheduled with a "normal speed" or "turbo speed". In the former case, the job will start at time s_j and finish at time $s_j + p_j$. In the latter case, the job will start at time s_j and finish at time $s_j + p_j/2$. That is, in the turbo mode, the speed of processing job j will double. However, you can run at most k jobs in the turbo mode. Your goal is to schedule jobs to maximize the total weight of completed jobs.

Design a dynamic programming algorithm for this problem. The algorithm receives arrays s_j, p_j, w_j and parameter k.

I. In what order does your algorithm sort jobs?

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

III. Describe the base case for your DP.

IV. Write the recurrence relation.

V. Explain why the recurrence relation is correct.

VI. Show how to solve the original problem (i.e., the problem of finding the most profitable schedule) using the solution to the DP.

Remark: The running time of your algorithm should be polynomial in n. You do not need to optimize the running time as long as it is polynomial.

Problem 2. You are going for a vacation to France. Before your trip, you chose n attractions you want to visit and picked a specific order in which you want to visit them. In your list, the coordinates of attraction i are stored in variable `point[i]`. The starting and ending points of your journey are the same. When you arrived to Paris, you realize that you cannot visit all attractions, because the maximum distance you can travel is `maxDistance`. So you would like to remove some of the attractions from your list so that the total length of the tour is at most `maxDistance`. Design and implement an algorithm that finds the maximum number of attractions you can visit. Note that

- The algorithm cannot change the order in which you visit attractions.
- Your trip must start and finish at location $(0, 0)$.
- The length of the entire trip must be at most `maxDistance`.
- The distance between two points `p` and `q` equals $\sqrt{(p.x - q.x)^2 + (p.y - q.y)^2}$.

Write the following function

```
MaxTour(const std::vector<Point>& points, double maxDistance)
```

The parameters of this function are `points` – attractions you plan to visit (each point has two coordinates x and y); `maxDistance` –the maximum distance you can travel. The function should return the maximum number of attractions you can visit. The running time of your program should not exceed several seconds (please, compile your code with `-O2` or `-O3` switch).

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_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!)
- `small_problem_set_5.in` and `large_problem_set_5.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 `MaxTour`. 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 Clang compiler, type
```
clang++ -std=c++17 -pedantic-errors problem_solver_5.cpp -O3 -o problem_solver_5
```
- If you use GNU C++ compiler, type
```
g++ -std=c++17 -pedantic-errors problem_solver_5.cpp -O3 -o problem_solver_5
```
- If you use Microsoft Visual C++ compiler, start `Developer Command Prompt` and type
```
cl /EHsc /O2 problem_solver_5.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_5 small` to run your code on simple problems and `./problem_solver_5 large` to run your code on hard problems. On Windows, type `problem_solver_5.exe small` and `problem_solver_5.exe large`, respectively. Make sure that executable is located in the same folder as files `small_problem_set_5.in` and `large_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! `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.
```

Please, test your code with the both problem sets (small and large). When your code is ready, submit file `student_code_5.h` on Canvas. Make sure that you are submitting the latest versions!