CPSC 4100
Design and Analysis of Algorithms
Homework 5 (100 points)
Due: Feb-25-2019, 11:59 PM (on Canvas)

Instructions:

NOTE: Please submit your work by April 26 on Canvas. This assignment is to be done individually; you can discuss the questions with your classmates, but you should code your solutions individually.

Code your solutions in C++. Your code must compile and execute on the department server cs1.seattleu.edu. If you do not have an account on the server, please let me know. Submissions not executing/compiling on CS1 will receive a zero.

If you have trouble following or solving the problem, please stop by or email me. I am happy to help!

1. In this assignment, you will write C++ program for solving the 0-1 Knapsack problem using dynamic programing. The problem is the following. A pirate pillaging a village finds n items. The i-th item is worth v_i dollars and weighs w_i kilograms, where v_i and w_i are integers. The pirate wants to maximize the total (dollar) value of the loot he takes, but he can carry at most W kilograms in his knapsack, for some integer W. Which items should he take? This is a 0-1 knapsack problem because for each item, the pirate must either take it or leave it behind; he/she cannot take a fractional amount of an item or take an item more than once. Your program should print the following:

I. The optimal (maximum) dollar value that can be carried in the knapsack for a given set of items. **(40 points)**
II. The items that should be taken for obtaining the maximum dollar value. **(45 points)**

Your program should accept only the following two **command line inputs** (before the program execution starts):

a. Knapsack capacity.
b. Input file name (and path if not in the current directory) that contains the value and weight information for all the items.

Format of the input file:

- The input file containing items and weight information will have two lines.
- First line contains the weights in kg; it starts with the label "Weight." Following the label would be a comma separated sequence of item weights.
- The second line contains the item values in dollars; it starts with the label "Value." Following the label would be a comma separated sequence of item values.

- You can assume the total number of entries in the first and the second line would be the same. The i-th entry in both the lines corresponds to the weight and value of the same item.
- However, you **cannot assume** anything about how many entries would there be. On reading the file, your program should count the number of integers in the first or the second line to determine the total number of items.

A sample input.txt file is provided with the assignment to get you started. Before submission, however, you should test your program on several input files---not only the one provided. For final grading, we will test the submissions on different test cases. For full credit, your program must pass them all. A program failing several of the test inputs will receive a very low score.

2. Prepare a "README.txt" file for your submission. The file should contain the following: (10)

a) Instructions for compiling and executing your program(s). **Include an example command line for the same**.
b) If your implementation does not work, you should also document the problems in the README file, preferably with your explanation of why it does not work and how you would solve it if you had more time.

3. You should also comment your code well. The best way to go about it is to write comments while coding. (5)

What should you submit?

Upload your submission as a zip file on Canvas. The zip file should contain:

1. All your code files and any other files that might be needed for executing your code.
2. README.txt.

Hints and Resources:

- Here is a reference on passing command line arguments in C++: https://www.learncpp.com/cpp-tutorial/713-command-line-arguments/
- The algorithm we discussed in class does not keep track of which subset of items gives optimal solution. To compute the actual subset, we can add an auxiliary Boolean table/array called "Keep [w][i]," which is 1 if we decide to take the i-th item in the call "Knapsack(c,i)" and 0 otherwise.