CS 3353 Spring 2024
Programming Homework 4
Dynamic Programming -- Forming a space exploration team

Due Date: 5/3 (Fri) 11:59pm. ABSOULTELY NO EXTENSIONS.

You are the Captain of the starship Enterprice of the Untied Federation of Planets. You have received an order to assemble a team to go on a mission where your team is "to boldly go where nothing has gone before".

For this exciting adventure, you have various position to fill for your team (e.g. doctor, engineer, morale officer etc.). We assume you have P different positions that you have to fill. For each position there is a total of k applicants. (k is the same across all positions). The following rules of selection applies:

- You do not want to manage egos (two people in the same position will bring conflicts), so you are only to take at most one person per position.
- You are NOT required to fill in all positions. If you do not fill a position, it will be fill by an AI ("Are you sure it is Intelligent") unit.
- To simplify the selection process, your superior has assigned three numbers for each applicant:
    - Id: an ID for each applicant (this is for bookkeeping only, and is not involved in the selection process).
    - Cost: the cost to keep the person in the team
    - Benefit: the benefit that each person will bring to team.

  Each of this two numbers (cost/benefit) is a positive integer. There is no relationship between the two numbers.
- You are given a total budget for B, where the total cost of the applicants cannot exceed. (You do NOT need to spend your whole budget, though). Also if you do not fill a position, the Al unit has 0 cost and 0 benefit.

Your goal is given, P, B, k and the list of applicants, select the applicants to form a team that give the maximum total benefit (which is the sum of each applicants benefits).

Task

There is a program available in Canvas (name prog4.cpp) which do the following

- Ask the user for the following information
    - The value of B, P and k (in that order)
    - Then it will read the list of all applicants.
        - It will read all applicants of position 1 first, then all applicants for position 2 etc.
        - For each applicant, it will read its id, cost and benefit in that order
    - I have a file call "prog4test.txt" that you can use to look at the format. Or you can run your program and redirect the file to standard input. (If you want to modify the main program so that it open the file and read it that is fine.)

- The program will store the information of the applicants in a vector of vector (think of it as a two-dimensional array) called alist, where each vector of alist is a vector that store the list of applicants for one position.

- It will then call a function
  pair<float, vector<Applicant> >pickTeam(int budget, vector< vector<Applicant> >& alist)

  which it takes the available budget and the list of all applications, and it will return a pair:

  - The first number is the optimal benefit that can be achieved
  - The second is the list of applicants chosen that leads to the optimal benefit (notice that it is just a vector of all the applicants chosen, we do NOT organize them by position).

- The program will then print the result and exit.

You are to implement that pickTeam() method to actually find the optimal value and the corresponding applicants. You MUST use a dynamic programming algorithm. You CANNOT use a recursive algorithm.

What to hand in

You will be provided with a set of files:

- prog4.h: Header file. Store the structure that represent an applicant.
- prog4.cpp: Implement the pickTeam() method
- prog4main.cpp: Main program

What you need to do is to hand in an updated version of your prog4.cpp, you should name it. Inside you need to implement the pickTeam() method, ***using dynamic programming***.

Hints

- This problem is a variation of the 0-1 knapsack problem.
- Remember for each position you can at most pick one applicant.
- However, you do not have to pick any applicant for each position.
- The whole function should take less then 60-70 lines to implement. (My version takes less than 70 lines, including empty line to keep the code “readable”).

Extra credit (30 points)

Implement the extra function pickTeam_2() in the prog4.cpp file. For this function, you allow the option for a maximum of 2 applicants to be selected for each position (you still have the option to pick 1 or none).