In this homework you will write 2 programs to solve a "Production planning problem", the first will use dynamic programming, and the second will use linear programming.

You are expected to work in teams of upto 4. Fewer are discouraged, but acceptable.

The submission deadlines are: for the dynamic programming, C++ program: Wednesday 4th Nov midnight. For the LP based program (almost certainly python): Sunday Nov 8 midnight.

1 The production planning problem

The problem is a slight modification of the production problem from DPV 7.1.2.

We are given the (integer) demand $D[m]$ for carpets for the next M months, numbered $m = 0..M-1$. Assume that the carpets demanded in each month are to be delivered at the end of each month. At the beginning of month 0, we have E employees. In addition we can hire and fire employees at the beginning of each month as necessary, at a cost $Hcost, Fcost$ per employee respectively. Retaining a previously hired employee for the next month has no cost. Each employee is paid a salary S per month and will make C carpets each month, in regular working hours. However an employee can also be expected to produce upto OTC carpets in addition as overtime, and for this a charge $OTPrice$ has to be paid per carpet produced during overtime.

The carpets that remain after satisfying the demand at the end of each month (and the demand must be satisfied) must be put into a warehouse at a charge of W per carpet per month.

One day 1 of month M we may hire or fire employees if necessary so that we have E employees with which we started. On day 1 of month M it is OK if we have undelivered carpets; assume that we give these away with no additional cost or income from them.

The input is given in the following format.

M
$D[0], \ldots, D[M]$
$E, Hcost, Fcost$
S, C
$OTC, OTPrice$
W

There are no commas in the input, they have been given above only for readability. In C++, only the order matters since you will read input using `cin`.

2 Implementation 1:

This is expected to be in C++. Please start working on this right away.

3 Implementation 2:

This is expected to be in Python, using the Pulp library. You can start working on this if you know Python. However, soon we will give a sample program that solves an LP using Pulp. It might be adequate to modify that to suit our problem.

4 Evaluation:

We will give some test data sets on which your program would be expected to run correctly as well as fast. We are hoping to set up an autograder for this.

In addition you could make up some test cases.

A small report is expected, in which you can put your observations, e.g. which method runs fast, is there a language effect, are particular kinds of instances solved better by one method.

There will be a viva, in which you will be quizzed about your contribution to the effort and also your understanding of the effort.