This assignment focuses on implementing the algorithms designed in the previous one. (So please refer to that assignment for many details.) The **goal** is to gain more experience in implementing and evaluating algorithms.

Questions

1. (150 pts.) Implement all your algorithms from the previous assignment. Test and document your work carefully and submit your packaged source code and supporting documentation.

2. (30 pts.) Conduct a brief experimental study of your implementation, measuring the running time for a suitable collection of inputs. Include your test code in your electronic submission, with suitable documentation.

3. (20 pts.) Summarize your experimental results by making effective use of charts and tables. Comment on how well the experimental results match the predictions based on your answer to Question 6 of HW02. Highlight any significant differences and explain them the best you can. Include these results, comments, and explanations as a single PDF file in your submission.

IO format Your program should read from *standard input* and write to *standard output*. The first token (whitespace-delimited) of the input is either `E`, indicating that the desired output is an *enumeration* of all journeys, with costs, or `M`, indicating that the desired output is a *minimum-cost journey* (and its cost) only. The next input token is the distance n. The remainder of the input has $2k$ integers, for some nonnegative integer k. Conceptually, these are k pairs of integers. If the first element of a pair is negative, then the pair represents the negated value and location (distance from start) of a pile of coins. Otherwise the pair represents the cost and jump-distance of a pogo stick.

For enumeration, your program's output should be one or more lines: The first line consists of just one integer r, which is the number of possibilities for JJ's journey with the given inputs. It is followed by r lines, where each line lists the cost of a journey followed by the sequence of pogo-stick distances used to make a journey. These r lines should appear in lexicographically sorted order. For minimum-cost computation, your program's output should be a single line for a minimum-cost journey, formatted as for the enumeration case. If there is no feasible journey, the output should be empty.

Example If the input is

```
E 5 3 5 5 10 2 1 2 3
```

it means JJ needs to cover 5 meters using pogo sticks that cover 5, 10, 1, and 3 meters, with jump-costs 3, 5, 2, and 2, respectively. In this case, the 10-meter pogo stick is useless. We can use the 5-meter one once, in just one way. That leaves the 1- and 3-meter pogo sticks.

We can use the 3-meter one no more than once. If we use it zero times, then all we have is the 1-meter stick, so there is only one way: $1+1+1+1+1$; if we use it once, we have to have two uses of the 1-stick, giving the possibilities $1+1+3$, $1+3+1$, $3+1+1$. We compute the journey costs in the obvious way. So the output in this case is (note lexicographic ordering):

```
5
3 5
6 1 1 3
6 1 3 1
6 3 1 1
10 1 1 1 1 1
```

If the `E` in the input is replaced by `M`, the desired output is `b 5` denoting a minimum-cost journey with cost 3 and a single 5-meter jump. For the input `M 5 -1 0 2 1 -10 1 2 3`, the desired output is `-5 1 1 3` or `-5 1 3 1` denoting of the two minimum-cost (maximum money making) journeys.

Submission: Submit your work electronically as well-packaged source code. Follow the submission procedure used for the previous assignment, replacing `hw01` with `hw03` in the obvious places.

Reminders Recall, from the previous assignment, policies on collaboration and the use of external resources. Ask for clarifications if anything is unclear. The suggestions in the previous assignment apply to this one too. Use the newsgroup for all questions and discussions unless the matter is private.