Name: _____

This goal of this assignment is to get some experience in thinking about an algorithmic problem, and in devising, implementing, and evaluating solutions. Use the class newsgroup for clarifications and discussions.

The main problem In one phase of the side-scroller game *JJ's Jolly Jumping Journey*, called *J5* for short, the objective is to guide the protagonist, *JJ*, to an exit door that is n meters away from JJ's initial position. JJ can move only by using a collection of k pogo sticks: $p_1, p_2, \ldots, p_k$. Each pogo stick p_i is a precision device that will move JJ exactly d_i meters toward the exit (unless it would overshoot, in which case JJ goes splat against a wall—to be avoided). The pogo sticks all work in just one (forward) direction. Each pogo stick may be used any number of times. (JJ carries them in a backpack when they are not being used.) We would like to enumerate the number of different ways JJ can get to the exit door, as a function of n and the pogo-stick distances $D = (d_1, d_2, \ldots, d_k)$. For simplicity, we assume that n and d_i are all integers and that the d_i are all distinct.

Questions

1. (1 pt.) Write your name in the space provided above.
2. (9 pts.) Devise an algorithm for solving the main problem described above. Describe your algorithm in English as precisely as possible. (Continue on next page.)

3. (10 pts.) Explain why your algorithm is correct.

4. (10 pts.) Provide pseudocode, using the textbook's style as a guide, for your algorithm. Include explanatory comments and outline a proof of its correctness.

5. (10 pts.) State and justify the running time of your algorithm as a function of the number n and sequence of numbers D.

6. (10 pts.) Repeat Question 5 for working space (memory use) instead of running time.

7. (100 pts.) Implement your algorithm. Test and document your work carefully and submit your packaged source code and supporting documentation.

8. (20 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.

9. (30 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 5. 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 input consists of a sequence of integers. The first integer is the distance n that JJ needs to travel. All the remaining integers are the pogo-stick distances d_i. 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 sequence of pogo-stick distances used to make a journey. These r lines should appear in lexicographically sorted order.

Example If the input is

```
5 5 10 1 3
```

it means JJ needs to cover 5 meters using pogo sticks that cover 5, 10, 1, and 3 meters. 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 $1+1+3$, $1+3+1$, $3+1+1$. So the output in this case is (note lexicographic ordering):

```
5
1 1 1 1 1
1 1 3
1 3 1
3 1 1
5
```

Suggestions Although we do care about the efficiency of the algorithm and the implementation, correctness is far more important. To that end, it is a good strategy to focus on devising a simple and correct algorithm and using it to complete all the questions above. Then, time permitting, one can revisit things trying to improve efficiency. A good way to get started is to work out some small examples by hand. After a few examples, you may notice some patterns that may suggest an algorithm. Another good exercise is thinking about the

boundary cases: When will there be no solutions for JJ's journey? Can we place an upper bound on the number of possible journeys?

Using other resources You are permitted, and encouraged, to discuss the problem and solutions at a high level with classmates on the class newsgroup. However, all the work you submit must be your own creation. In particular, all code you submit must be your own work and you must make sure that you can explain how it works in detail. (An arbitrary subset of the class will be called upon to demonstrate this knowledge in person.) Any help you receive, be it from persons, books, papers, videos, Web sites, or other resources, must be very prominently noted in your submission; failure to do so amounts to academic dishonesty. (See syllabus.)

Submission: Submission consists of (1) a hardcopy of this assignment with answers filled in neatly and (2) a well-packaged electronic submission with answers to the programming components. Refer to the syllabus for submission instructions. Name the electronic submission using the template

`cos350-hw01-`*`lastname`*`-`*`firstname`*`-`*`pqrs`*`.jar`

where *`lastname`* and *`firstname`* are replaced by the obvious and *`pqrs`* is replaced by a 4-digit string of your choosing. The submission should be designed so that the command `jar xf cos350-hw01-`*`lastname`*`-`*`firstname`*`-`*`pqrs`*`.jar` results in the creation of a directory `cos350-hw01-`*`lastname`*`-`*`firstname`*`-`*`pqrs`*. In that directory should be all the source code (organized in further sub-directories as needed) as well as a README file with the usual semantics. *Do not submit any kind of non-source files* (executables, object files, class files, etc.). Be especially careful if you use IDEs such as Eclipse for your work because the packages created by them often contain compiled, not source files.