CSC 495 – Problem Set 10 – Due Tuesday, April 3

Problem 10.R1: Smoking Discs

Required Problem
Points: 50 points

Background

Stefan Banach was a very influential mathematician in the early 20th century — he has many serious mathematical concepts named after him, but also a simply-stated puzzle known as "Banach's matchbook problem." Imagine that a pipe smoker put two matchbooks in his pocket, both starting with N matches when new. Each time he lights his pipe, the smoker fishes in his pocket and pulls out a random matchbook — he always uses one and only one match, and then returns the matchbook to his pocket. The question is this: When he runs out of matches in one matchbook, how many matches are left (on average) in the other one? It turns out that the two matchbook problem can be analyzed exactly, but the problem can be generalized in different ways that make it more difficult to analyze. Note that while mathematical puzzles are often stated just for fun and for pushing the boundaries of what can be solved, many times they turn out to have unexpected applications — as a perfect example, note that the title of the following computer science research paper: "The generalized Banach match-box problem: Application in disc storage management" (by Krzysztof Goczyla).

When problems are difficult to solve analytically, we can often approximate answers using Monte Carlo simulations. We can simulate the Banach match box problem 1,000,000 times, average the results together, and get a good estimate of the actual expected value. When estimating an expected value through simulations, the speed at which this estimate converges to the actual expected value depends on the distribution (and most importantly on the variance of the distribution — look up "Chebyshev's Inequality" if you're curious about this). This can be studied and there are good ways of estimating error, but for this problem set we will take an unsophisticated strategy: pick a time limit for our simulation (around 30 seconds) and run as many simulations as possible within that time limit. This time bound is large enough to get a decent estimate for the problems we are interested in (assuming you write reasonably efficient code) — if you want to get a feel for how accurate this estimate is you can run the test several times and see how much the computed estimate varies. For this problem, we will simulate the k-matchbook problem.

The Problem to Solve

You are given two integers, k representing the number of matchbooks, and n giving the number of matches is a new matchbook. You should output an estimate of the average number of matches left when one matchbook runs out of matches (this is the total number in all of the $k - 1$ other matchbooks combined). You should run as many simulations as you can so that the total simulation time takes between 15 and 30 seconds. You are guaranteed that $k \leq 10$ and $n \leq 100$. The submission server will judge your program as correct as long as you are within 1% of the actual expected value.

Hints and Techniques

There are several ways you could figure out how many simulations to run within the target time range, but consider this as a particularly good idea: Recall the earlier problem set where you timed specific sections of code. Now consider running a certain number of simulations, seeing how long it takes, and if you haven't reached the minimum time desired you can double the number of simulations. If you do this right, you can hit any target range of the form "$t, \ldots, 2t$ seconds (like 15 to 30 seconds). Alternatively, instead of doubling every time, you could try to compute the number of iterations required once you have a decent estimate of how quickly your code is processing each iteration.

Note that you *could* check the time after every iteration, and stop once the desired time is reached. However, that turns out to be a really bad idea: there is a non-trivial amount of overhead involved in checking the time, so to avoid really slowing down your code you should try to check the current time as little as you can get away with.

Input and Output

Input consists of the two integers k and n, representing the number of matchbooks and the number of matches that each matchbook starts with, respectively. Your output should be a single line containing the estimate of the expected number of matches left at the time that the last match in a matchbook is used (a floating point number).

Sample Input

```
4 100
```

Sample Output

```
40.390462420654295
```

Problem 10.R2: Swinging with the Stock Market

Required Problem
Points: 50 points

Background

Consider a system which evolves in the following way: the system has a "state," which changes in a sequence of steps, with the next state chosen according to some probability distribution (the probability distribution can depend on the current state or be independent of it). As the state evolves over time, we can ask all sorts of interesting questions — some of which can be analyzed, and some of which we must simulate.

For example, you could play the following game: Two players each roll a 6-sided die at each step, and the player with the lower number pays the player with the higher number $1 (if there's a tie, no payment is made). You are one of the players, and your state is the amount of money that you have. With probability 1/6 the amount of money you have doesn't change, with probability 5/12 you lose a dollar, and with probability 5/12 you gain a dollar. We can describe this state update function with the following table (where probabilities are rounded):

Probability	Change
0.4166667	-1
0.1666666	0
0.4166667	1

We can ask questions like the following: if we play this game for 50 steps, what is the expected smallest balance (or largest loss) that we experience over those 50 steps, and what is the expected largest balance (or largest winnings) that we experience. It turns out that the expected maximum win or loss over 50 steps is between 4 and 5.

This kind of modeling is referred to as a "random walk," and there are many examples of more complex random walks. For example, you can look at the value of the stock market, as measured by the Dow Jones Industrial Average (DJIA), as a random walk. If we had an accurate model of how the DJIA changes from day to day, we could ask questions such as: what are the expected least and greatest values over the next year? In this problem we just consider random walks in which a value is either added to or subtracted from at each step, so what we would model is not the actual increases and decreases in the DJIA, but increases/decreases in the *logarithm* of the DJIA (can you figure out why?).

The Problem to Solve

You will be given a state update table, as shown above (with m rows), and a number of steps (n). You should should repeatedly simulate this system for the required number of steps, computing estimates of the minimum and maximum value achieved by the system during that time period. As in the previous problem, run the simulation so that it takes between 15 and 30 seconds. You are guaranteed that $m \leq 20$ and $n \leq 300$. Furthermore, the probabilities will always add up to 1.

Input and Output

Input consists of a line containing m (the number of possible increments) and n (the number of steps that the system runs). Your output should consist of two lines, containing estimates of the expected minimum value and maximum value achieved by the system. The sample below shows results for the dice game described in the "Background" section.

Sample Input 1

```
3 50
-1 0.4166667
0 0.1666666
1 0.4166667
```

Sample Output 1

```
-4.678677
4.678387
```

The following sample input/output is provided for fun — no claim at accurate stock market prediction is made! The distribution given below is based on daily closing prices for the Dow Jones Industrial Average over the past 20 years. First, prices were replaced by the base 10 logarithm of the price, multiplied by 1000. Then an approximate distribution of daily change in these values was obtained (simplifying into 9 increment/decrement "buckets"). The result is shown below:

Sample Input 2

```
9 200
-9.4290 0.0844
-3.7553 0.1354
-1.9632 0.0854
-0.9722 0.1014
-0.0075 0.1264
0.9711 0.1185
1.9663 0.0953
3.7564 0.1614
8.9224 0.0918
```

Sample Output 2

```
-34.012125
65.284648
```

Here's an interpretation of these results: The expected min of -34.0 means that the expected minimum price over the next 200 trading days is $10^{-0.034} \approx 0.925$ of the current DJIA, and the expected max is $10^{0.0653} \approx 1.162$ times the current DJIA. Given today's DJIA of 13,252, the expected range over the next 200 trading days is therefore about 12,258 to 15,399. This all assumes that the model is an accurate predictor, and we have an average year (pretty big assumptions).