(a) You are playing the following game: you roll a fair six-sided die, and then you throw a certain number of coins and get $1 for every coin that falls heads. The amount of coins you throw is the square of the value that appears on the die. (For example, if the die comes out 4, you throw 16 coins). What is your expected profit?

(b) The same game as in (a) except that if you roll a 6, then in addition to throwing 36 coins and collecting $1 for every head, you also get to play the game again. So as long as you are getting sixes, you keep playing the game. What is your expected profit in this game?

2.) Consider the following program which does a randomized “binary-like” search.

Input: A,k,left,right; where A is a sorted (in increasing order) n-element array indexed 1 through n, and k is the number in the array that we are looking for. left and right are indexes between which k is located. Initially, left should be 1 and right should be n.

Output: print i such that A[i] = k.

randSearch(A, k, left, right)
    j ← random number from the set \{left, left + 1, \ldots, right\}
    if A[j] = k
        print(j)
    else if k < A[j]
        randSearch(A, k, left, j - 1)
    else
        randSearch(A, k, j + 1, right)

Suppose we run randSearch(A, k, 1, n) where k happens to be the smallest number in the array. What is the expected running time of the algorithm in this case?
3.) Suppose you have $n$ distinct pairs of socks in your sock drawer ($2n$ total socks). The socks are not matched up, so they are distributed completely randomly in the drawer. Every morning, you reach into the drawer, pick out a sock at random and put it on your bed. Then you reach into the drawer again, take another random sock, and put that one on your bed. If the two socks match, then you’ve got a pair, and you’re done. If not, then you reach in and get another random sock, and so on. You stop when the sock from the drawer that you just picked up matches with some sock already on your bed. We want to figure out what is the expected number of socks you have to take out of the drawer until you get a matching pair.

(a) Let $T(j)$ be the expected number of times you need to pick a sock to get a match if there are $j$ socks remaining in the drawer (so you already picked $2n - j$ without a match). Write a recurrence for $T(j)$, but do not solve it.

(b) Explain why $T(n)$ has to be 1. (So, a way to check if your recurrence from part (a) is incorrect is to plug in $n$ for $j$ and see if you don’t get 1).

(c) This recurrence seems a bit hard to solve on paper. Instead, write a computer program in any language you want (on the homework, write the pseudocode for your program) to compute this recurrence. Your program should take as input the number of pairs of socks that you start out with, and output the expected number of socks that you will need to pick in order to get a matching pair. Using your computer program, compute the answer for 10, 100, 1000, and 10000 pairs.

(d) Using your output, can you see what the expected number of socks you need to pick is, as a function of the total pair of socks? If not, then square the expected number of socks you need to draw to get a match and divide that by the total pairs of socks. You should see the quotient approach a certain familiar number. What is it? Now can you answer what the expected number of socks that you need to draw to get a match?