EE1301: Introduction to Computing Systems
Fall, 2018
Department of Electrical and Computer Engineering
University of Minnesota

Homework #2

Instructions:

Follow all of the instructions at the beginning of Homework #1.

Problem 1.

Write a program that requests the user to enter an integer between -50 and +50. Your program must check that the entered value is within this range and repeatedly ask the user to enter it again if it is not. For positive numbers, print a pattern that has '1' on the first line, '22' on the second line, '333' on the third line, and so on up to the value of the number entered. For negative numbers, print the pattern in the opposite direction, that is, with the largest number printed first down to 1. Repeat this process until the user enters 0. For example:

```
Enter an integer from -50 to 50:  4
1
22
333
4444

Enter an integer from -50 to 50:  -4
4444
333
22
1

Enter an integer from -50 to 50:  0
Goodbye.
```

Problem 2.

Write a program that prompts the user to enter a non-zero floating-point value, call it `x`. Then repeatedly prompt the user to enter additional floating-point values, which we will call `a`. After each value of `a` is entered, print the result of computing the function f(n) = a * x^n + f(n-1), where f(n-1) is the value computed in the previous iteration. Also print the value of n. Assume that f(0) = 0. You can use only the built-in math functions +, -, *, and /. No functions from the math library are allowed. For example:

```
Enter a non-zero value for x:  5.0
Enter a floating-point value for a:  3.2
The value of f(1) = 16
Enter a floating-point value for a:  -6.81
The value of f(2) = -154.25
Enter a floating-point value for a:  4.45
The value of f(3) = 402
```

... and so on. Terminate the program when the user enters -999999. Minimize the total number of multiplications your program needs to do by saving and reusing the results from the previous iteration.

Problem 3.

Write a program that simulates a game in which four wheels spin and randomly stop at a number between 1 and d, where your program prompts the user to enter a value for d. If the four numbers match, print the numbers and "Eureka!". Otherwise, print the numbers and "You lose." Your program should continue the process until the value -1 is entered for d. For example:

```
How many values do you want on each wheel?  4
The wheels spin to give:  3 2 4 3.  You lose.
How many values do you want on each wheel?  3
The wheels spin to give:  2 2 2 2.  Eureka!
How many values do you want on each wheel?  15
The wheels spin to give:  14 8 3 12.  You lose.
How many values do you want on each wheel?  -1
OK, goodbye.
```

An example program demonstrating how to use the `rand()` and `srand()` functions to generate random numbers is shown at this web site: http://www.cplusplus.com/reference/cstdlib/rand/. You can use these functions to randomly generate the values to simulate the spinning of the wheels. You should also define your own functions to make your program easier to write and easier for someone else to understand.

Problem 4.

In this problem, you will extend the spinning wheel game from the previous problem to simulate the probability of winning for different combinations of the number of wheels and the range of values on each wheel. Let w = the number of wheels to be simulated (w=4 in the previous problem). On each wheel, the range of values is 1 to d. Write a function called `spin_the_wheels(d, w)` that takes integer values d and w as input parameters. Your function should return 1 if all w values match from the simulated spin of the wheels. Otherwise, it should return 0.

Now put this function within a for loop that calls the function n times and counts the number of times that the function returns 1. Call this count of the number of wins, m. Then the ratio m/n is a simulated estimate of the probability of winning with the given values of d and w. The larger the value of n, the better the estimate will be.

The next step is to enclose this loop within another loop that varies the value of d, and then enclose both of those loops into another loop that varies the value of w. This process will produce estimates of winning for a range of values of d and w.

Write a program that performs the above simulation experiment for w = {3, 4, 5, 6} and d = {9, 12, 15, ..., 27}. Use n = 1,000,000 for each experiment. Your program should print the values of w and d followed by the simulated win probability and the theoretical win probability. For example:

```
w=3, d=9:  Simulated probability = m/n = XX%.  Theoretical probability = XX%.
w=3, d=12:  Simulated probability = m/n = XX%.  Theoretical probability = XX%.
w=3, d=15:  Simulated probability = m/n = XX%.  Theoretical probability = XX%.
etc.
```

The X, m, and n values above should be replaced with the values computed by your program (the variables used to compute the percentages should be of type `double;` the counts should all be of type `int`).

To compute the theoretical win probability, we need to divide the total number of ways to win by the total number of possible combinations. A win is defined to be the case when all w wheels have the same value. Since there a d unique values on each wheel, there are d ways in which you can win. The total number of combinations possible with w wheels is $(d \times d \times ... \times d) = d^w$. So the theoretical probability of winning is d / d^w.