PS2

`Q1`

In the land of Puzzlevania, Aaron, Bob, and Charlie had an argument over which one of them was the greatest puzzler of all time. To end the argument once and for all, they agreed on a duel to the death. Aaron was a poor shooter and only hits his target with a probability of 1/3. Bob was a bit better and hits his target with a probability of 1/2. Charlie is an expert marksman and never misses. A hit means a kill and the person hit drops out of the duel. To compensate for the inequities in their marksmanship skills, it is decided that the contestants would fire in turns starting with Aaron, followed by Bob, and then by Charlie. The cycle would repeat until there was one man standing. And that man would be remembered for all time as the Greatest Puzzler of All Time. An obvious and reasonably strategy is for each man to shoot at the most accurate shooter still alive, on the grounds that this shooter is the deadliest and has the best chance of hitting back. Write a program to simulate the duel using this strategy. Your program should use random numbers and the probabilities given in the problem to determine if a shooter hits the target. You will likely want to create multiple subroutines and functions to complete the problem. Once you can simulate a single duel, add a loop to your program that simulates 10,000 duels. Count the number of times that each contestant wins and print the probability of winning for each contestant (e.g. for Aaron your program might output "Aaron won 3595/10000 duels or 35.95 %").

`Q2`

An array can be used to store large integers one digit at a time. For example, the integer `1234` could be stored in the array a by setting `a[0]` to `1` , `a[1]` to `2` , `a[2]` to `3` , and `a[3]` to `4` . However, for this exercise you might find it more useful to store the digits backward, that is, place `4` in `a[0]` , `3` in `a[1]` , `2` in `a[2]` , and `1` in `a[3]` . In this exercise you will write a program that reads in two positive integers that are `20` or fewer digits in length and then outputs the sum of the two numbers. Your program will read the digits as values of type char so that the number `1234` is read as the four characters `1` , `2` , `3` , and `4` . After they are read into the program, the characters are changed to values of type int . The digits will be read into a partially filled array, and you might find it useful to reverse the order of the elements in the array after the array is filled with data from the keyboard. (Whether or not you reverse the order of the elements in the array is up to you. It can be done either way, and each way has its advantages and disadvantages.) Your program will perform the addition by implementing the usual paper-and-pencil addition algorithm. The result of the addition is stored in an array of size `20` and the result is then written to the screen. If the result of the addition is an integer with more than the maximum number of digits (that is, more than `20` digits), then your program should issue a message saying that it has encountered "integer overflow." You should be able to change the maximum length of the integers by changing only one globally defined constant. Include a loop that allows the user to continue to do more additions until the user says the program should end.

Turn In

- Make and submit a zip file(`<your_full_name>_PS2.zip`) which includes the following:
  - Source code of `Q1`: `q1.cpp`
  - Source code of `Q2`: `q2.cpp`
  - Run `Q1` and `Q2` and attach screenshots(in `jpg` format, not exceeding `300kb` each) which show that your programs are running.
  - At least `1` screenshot for each question.