**CS 575 Homework**

Here are several little programs I want you to write, using loops. Some of the programs require the use of the die and restore functions. These aren't C++ functions from the standard library, they're just a couple simple functions that I wrote, which are posted on Moodle. You may copy them into your program and use them as is, or you write and use your own versions of them if you want to and know how.

As usual in this class, we'll ignore the possibility that huge user input might cause an overflow.

As usual in this class, we'll ignore the possibility that the user might type a negative value when the program is inputting an unsigned (in which case Microsoft C++ does something weird and unhelpful).

As usual in this class, I don't care about formatting niceties like how many digits appear after the decimal point.

As usual in this class, you may not declare any global variables. (Global constants are fine, though.)

For this assignment, for problems that don't mention prompts, I don't care whether you output a prompt or not.

A. Multiples of k less than n: for, while, and do-while

Input values into an unsigned that I'll call k (you can call it whatever you like) and another unsigned that I'll call n. If the input fails, complain and die. If k is 0, complain and die.

The program's job is to output all the non-negative multiples of k that are less than n, on one line, and to do that three times. Each of the tree times, the multiples (if there are more than one of them) are separated by commas, without a comma before the first number or after the last number.

You will generate the first line of output with a while loop.

You will generate the second line of output with a for loop.

You will generate the third line of output with a do-while loop.

An example run of your program might go as

**5 40**

0, 5, 10, 15, 20, 25, 30, 35

0, 5, 10, 15, 20, 25, 30, 35

0, 5, 10, 15, 20, 25, 30, 35

Another example run of your program might go as

12 0

(In this case, we got three blank lines, since there are no non-negative multiples of 12 that are less than 0.)

B. Input an unsigned that I'll call k. (c&d on input failure). Then input k double values. (c&d on input failure.) Output the sum and product of the numbers. (If k is zero, then 0 doubles will be input; the sum of 0 numbers is 0, and the product of 0 numbers is 1)

An example run of your program might go as

**4 2 3 4 .5**

sum is 9.5, product is 12

Another example run of your program might go as

**0**

sum is 0, product is 1

C. Input unsigneds, non-# to quit. Output the number of times an odd number was entered.

An example run of your program might go as

**5 10 20 1 1 50 1234 blah**

3

D. Input doubles, non-# to quit. If the number of doubles is 0, just output

no data

Otherwise, output the average of the values entered.

An example run of your program might go as

**1 2 3 4 blah**

2.5

Another example run of your program might go as

**blah**

no data

E. This program's job is to input an unsigned that is a multiple of 7, and output that number.

If the user enters a non-#, complain (don't die) about the input being a non-#, and try again (after calling the restore function).

If the user enters a # that is not a multiple of 7, complain about that (don't die) and try again.

Keep going until the user finally enters a number that is divisible by 7, then output that number.

An example run of your program might go as

**blah**

That's not even a number.

**100**

That's not divisible by 7

**1**

That's not divisible by 7

**hello**

That's not even a number.

**105**

Thanks for the 105

F. This program's job is to input doubles, non-# to quit.

If the number of doubles is 0, just output

no data

Otherwise, output the largest # input.

An example run of your program might go as

**8 3 -10 -4 0 27.1 -1234 blah**

27.1

Another example run of your program might go as

**blah**

no data

G. Write a program that adds all the numbers from 1 to 1000000 and outputs the result.

No input. Warning: the answer to this math problem is too big to fit in an unsigned or an int.

H. My secret password is "Gandalf" (don't tell anybody). Write a program that asks the user for my password and inputs a word. (For this program, you may assume without checking that the input is successful.) If the word is "Gandalf" (case sensitive), output a message to the effect that the user is being given computer access. If the word is not "Gandalf", output a message to that effect, and let the user try again. If the user eventually gets it right, output a message to that effect. Otherwise, continue telling the user that he's wrong and letting him try again until the user makes 5 wrong guesses. Use a loop. After 5 wrong guesses, output a message to the effect that the user is being denied computer access.

An example run of your program might go as

Password: **Gandalf**

Ok, you now have access to Bob Hart's checking account.

Another example run of your program might go as

Password: **Frodo**

Nope, try again: **Pippin**

Nope, try again: **Gandalf**

Ok, you now have access to Bob Hart's checking account.

Another example run of your program might go as

Password: **Frodo**

Nope, try again: **Pippin**

Nope, try again: **Sauron**

Nope, try again: **Bifur**

Nope, try again: **Bofur**

You are obviously a criminal. Homeland Security has been notified.