Exercise 04

For this exercise, you get to try out some of the unsigned integer types available in C, types we didn't have in the Java. On our course homepage, you'll find a partial implementation of a program called **digits.c**, along with some sample input and output files you can try your program with. You can also download them to you AFS file space with the following curl commands.

```
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise04/digits.c
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise04/input-1.txt
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise04/expected-1.txt
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise04/input-2.txt
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise04/expected-2.txt
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise04/input-3.txt
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise04/expected-3.txt
```

This program's job is to read two unsigned long values, low and high, from the terminal. Using the unsigned long type will let our program work with larger (positive) values than we could with any of the other integer types. After reading low and high, your program will read an unsigned int giving a target value to look for (more about this later). To read in an unsigned long value, you can use the "%lu" conversion specifier (the l is for long, and the u is for unsigned). The last input value is just an unsigned int, so you can read it with "%u". If you're not able to read these values successfully (say, if the user types in garbage), your program should exit unsuccessfully with an exit status of EXIT_FAILURE. Remember, the return value from scanf() tells you how much input it was able to read successfully.

Given valid inputs, your program is supposed to check all the integer values from low up to and including high. For each value, you're going to add up the numeric values of all its digits. For example, a number like 274 has digits 2, 7 and 4. If you add these up, you get 2 + 7 + 4 = 13. If you find a value with a digit sum equal to the target (the third input value), you'll print out that value on a line by itself. You can print unsigned long values using the same conversion specification you use to read them in.

As you complete this program, most of your code will go in your main function. There's a little bit of code in main() already, along with some comments telling you what code you'll need to add.

The job of adding up the digits in a number is done by a function, digitSum(), that you get to implement. Notice that this function takes an unsigned long as a parameter and returns an unsigned int as its result. Hint (if you need it): To add up the digits in a number, you can use the mod and division operators to work your way from the low-order digit (the ones place) to the highest-order digit. You can mod by ten to get the value of just the low-order digit, and you can divide by ten to discard the low-order digit and slide the rest of the digits down by one place. Your digitSum() function can loop, doing these two operations over and over until it has extracted the values of all the digits.

Sample Execution

Once your program is complete, you should be able to run it as follows. The user input is in bold, and the rest is the program's output. Here, we're checking all the values from 0 up to 100, reporting any values with a digit sum of 9.

```
$ ./digits
0 100
9
9
18
27
36
45
54
63
```

```
72
81
90
```

Your program should work with integer values that are too big to fit in anything but an unsigned long

```
$ /digits
9999999999999997999 9999999999999999997
169
9999999999999997999
9999999999999998899
9999999999999998989
9999999999999998998
9999999999999999799
9999999999999999889
9999999999999999898
9999999999999999979
9999999999999999988
9999999999999999997
```

If the user enters garbage, your program should exit unsuccessfully as soon as it sees the invalid input.

```
$ ./digits
123456789 abcdefghijklmnop
$ echo $?
1
```

When you're done, submit the source for your digits.c program to the exercise_04 assignment on Moodle.