CS 100 Lab Three – Spring 2019

Create a directory called **lab3** on your machine using **mkdir lab3** and move into that directory with **cd lab3**

Complete the following programs. Make sure to prompt the user for any input needed by the program. **Whenever possible, please end the prompt with a newline, which will make the output more readable by the grader.**

1. Name this program **random.c** – This program uses the pseudo-random number generator to generate a specific number of random numbers (integers) in the range of **0** through **99**. The program first prompts for the number of random numbers to generate, then generates that many random numbers, and determines how many different numbers out of 100 numbers (**0** through **99)** were generated. To use the pseudo-random number generator, you first need to call the **srand** function declared in <stdlib.h> once to set the seed. Usually you use the current time as the seed, but for ease of grading, please call **srand(0)** to use 0 as the seed. Then use **rand()%100** repeatedly to get random numbers in the range of 0 to 99. Please note different compilers may use different algorithms to generate random numbers. If you test your program on the **cs-intro** server, you should get the following results (from three executions). If you run your program on a Mac or PC, you may get different results.

```
How many random numbers to generate:
1
1 out of 100 numbers were generated after 1 attempts
```

```
How many random numbers to generate:
10
9 out of 100 numbers were generated after 10 attempts
```

```
How many random numbers to generate:
250
93 out of 100 numbers were generated after 250 attempts
```

Hints:

- Use an array of 100 elements to store the number of times you've seen the numbers 0, 1, 2, … 99.

2. Name this program **missing.c** – This program reads in a series of words. All words consist of only lower-case letters ('a' through 'z'). None of the words entered will be longer than 50 letters. The program reads until ctrl-d (end-of-file), and then prints out all the lower-case letters ((in ascending order) that were missing in the input or a statement indicating the input has all the letters. Two executions of the program are shown below.

```
Enter your input:
the quick brown fox jumps over the lazy old dog
Your input contains all the letters
```

```
Enter your input:
alabama crimson tide
Missing letters: f g h j k p q u v w x y z
```

Hints:

- Use an array of 26 elements to store the number of times you have seen the characters 'a' through 'z'.
- 'a' is actually an integer of value 97 (its ASCII value). If you subtract 'a' (or 97) from a lower-case letter, you will get a number in the range of 0 to 25.
- The input could contain multiple lines. To test this program, it is a good idea to use vim to create a file (**datafile** for example) to contain all the lines. Then test the program with input redirection as below. **./a.out < datafile** (or **./a.exe < datafile**)

Submit your lab

First, on your local machine, compress your **lab3** directory into a single (compressed) file, i.e. **lab3.zip**. Second, once you have a compressed file named **lab3.zip**, submit that file to Blackboard.