Lab 3: Generating and Analyzing Random Strings

Scientists routinely need random data for experiments and simulations. Natural events appear to be random, and in order to study and simulate them, scientists use randomization. Computer scientists especially need to know how to generate random data since they are the ones who are often called upon to design the software. It is sufficient to be able to generate ***random numbers***, because if we have a sequence of random numbers, we can generate random data of other types from it.[1]

There is a contradiction in the idea that a computer program can generate random numbers, because if a computer program generates anything, it is the result of an algorithm and is therefore not random! A computer program can generate numbers *that look random* because no mere mortal can detect a pattern in them, nor could another very smart computer, even if given a very long time. Such "random looking numbers" are called ***pseudo-random numbers***.

In this exercise you will write a program that generates pseudo-random text strings of a specific form, consisting only of the lowercase letters 'a', 'c', 'g', and 't'. In other words, the program will generate a string that contains just these letters, but the order of the letters and the quantity of each type of letter is pseudo-random. Such a string will be called a ***pseudo-random DNA string*** here. Once you have successfully completed this first program, you will enhance it a bit.

Exercises

Problem 1. Write a program that begins by prompting a user to enter a positive integer N. You can assume that the user will always enter an integer, but the program must detect if the user entered a negative integer or zero, and if she did, it should exit with an error message. The program then prompts the user to enter the name of file into which it will store the generated DNA string. The program then generates a pseudo-random DNA string of length N and writes it into a file with the chosen name, in its current working directory.

Problem 2. Once the program from *Problem 1* is working correctly, and only when it is working correctly, you will make a small modification to it. In addition to writing the pseudo-random DNA string to the file, it must display on the terminal window the fractions of 'a's, 'c's, 'g's, and 't's in the string. For example, if the string is length 50 and it contains exactly 20 'a's, 10 'c's, 5 'g's, and 15 't's, then it would display something like:

```
a: 0.4000 ; c: 0.2000 ; g: 0.1000 ; t: 0.3000
```

Notice that the sum of these fractions must be 1.0. The program should display the fractions with four digits of precision to the right of the decimal point.

[1] In fact it is a theorem of computer science that all data, however complex it may appear, is representable as a set of non-negative integers.

Testing Your Work

Before you submit your program, you have to make sure it is correct. It is not easy to test a program that generates random data, because in order to test, one needs to reproduce a result repeatedly. The `srand()` function is useful here. If `srand()` is given the same value, the sequence of random numbers produced by `rand()` will be the same. You should make sure also that the length of the string is correct, that the data looks random, and that your fractions add up to 1. If you give `srand()` a new value in each run of the program, then you should see a different DNA string each time.

What to Submit

Unlike the previous lab assignment, ***you will submit just one program***. If you finished the second problem and it is working correctly, submit that program and not the first. I will not look at the first program and it will not count toward your grade.

If you did not get the second program working, but the first was working, you may either submit the first, or submit the partially working second one. The first program is worth only 6 out of 10 points. The second is 10 points. If the second is almost working, and you think you will lose fewer than 4 points on it, submit it. If you know it needs much more work, submit the first instead.

Submit your program, whichever it is, by the end of today's lab, i.e., before the end of the class at 2:00 P.M. The instructions are just like the previous lab's:

1. Create a directory in `/data/biocs/b/student.accounts/cs135_sw/cs136labs/lab03/submissions` whose name is your *username*. For example, I would create the directory `sweiss`.

2. Copy your program, which should be named either *username*`_lab03a.cpp` or *username*`_lab03b.cpp` to this directory. You will lose 5% of the grade if you misname the file!

3. Change the permission on the directory that you created so that no one else can read or modify it. You do this with the commands

   ```
   $ cd /data/biocs/b/student.accounts/cs135_sw/cs136labs/lab03/submissions
   $ chmod 700 username
   ```

Do not submit executable files. Remember to document your code (both preamble and comments in the code) and make it easy to read. Your work will be judged using the rubric I outline in the Programming Rules document.

There are absolutely no extensions to the deadline. You can submit a revised version of the program by 10:00 P.M. on Thursday, Sept. 13 for partial credit. If you do, you must name it with a different name than what you first posted, e.g., *username*`_lab03b_v2.cpp`. and put it in the same directory as the original. It must be a *revision* of the original file, with only fixes to things that were not working before. I will adjust the grade based on how well the first version worked and how many changes you made.