CS 241L - Data Organization

Fall 2022

Programming Assignment #4
Total points: 100
Due date: September 27, 2022

In this assignment, you will write a simple C program that includes processing input and using control structures.

1 Extract files from tarball:

Log ino the cs machines. Create a directory for this assignment as follows:

```
$ cd cs241 ; $ mkdir prog04
```

```
$ chmod 700 prog04 ; $ cd prog04
```

The necessary files are given to you as a compressed tarball. In order to extract the files from this tarball use the command:
`$ tar -xvf programming_assignment_4.tar.gz`.
This will create the following seven (7) files in your current directory: `dnaInput_1`, `dnaInput_2`, `dnaInput_3`, `dnaInput_4`, `outputFile_1`, `outputFile_2`, and `outputFile_3`.

2 Description of the assignment

DNA is the basis of life here on earth. Every living creature is described by their DNA, which is a very, very long sequence of the bases A, C, G and T. Bioinformatics is a new discipline that uses the power of computers to analyze biological data, to discover new insights about the mysteries of life. For this assignment, you will complete a very simple bioinformatics experiment.

The input for your program will be a text file consisting of a sequence of DNA bases. Each base can appear as either a upper-case or as a lowercase letter (i.e., either 'A' or 'a' is legal). Your program must read the file, character by character, counting the total number of occurrences of each type of base. Your program will then output, for each base, what percentage of the DNA sequence consists of that base. For example, if the input is the sequence: **ACGGGTCGAC** then the output will be **exactly** these five lines of text:

The DNA sequence has 10 bases

- 20.00% of the bases are A
- 30.00% of the bases are C
- 40.00% of the bases are G
- 10.00% of the bases are T

The output must display the percentages with **exactly** three places to the left of the decimal point (to allow for 100%). The output must display the percentages with **exactly** two places to the right of the decimal point. The percent signs in the output must all be vertically aligned.

If your program successfully completes its task, then the program should return an exit status of 0 to the operating system. The input data file may contain any arbitrary "white space" (tabs, blank spaces, newlines) intermixed with the characters representing the DNA bases. These chars should not be counted. If the input data contains any other symbols, then your program should abort processing, and return an exit status of 1 to the operating system.

Since there are four kinds of bases, this program is a very natural place to use a switch statement. Your program **must** incorporate a **switch**. Your program **must** read from the standard input (stdin), and not from any other file.

Create a `.c` file for this assignment and name it using your last name and the initial of your first name, like this: `lastName_initialFirstName_DNA.c`

To compile your file you should use the command:

```
$ gcc -std=c99 -pedantic -Wall lastName_initialFirstName_DNA.c
```

I have provided four input files and the corresponding correct output files for three of them; one of the input files has other symbols. If you develop a program called with the requested name for this assignment, then you can test your program using the first test data file as follows:

```
$ gcc -std=c99 -pedantic -Wall lastName_initialFirstName_DNA.c

$ ./a.out < dnaInput_1 > myOutput_1

$ diff myOutput_1 outputFile_1
```

You may repeat this process for all the other dnaInput_i files where i may be 2, 3 or 4. The output of your program should be **char-by-char** identical to the given outputFile files, for those input files with correct input. The **diff** command will display any places where the two argument files are different. If the **diff** command is silent, then you know the two files are the same. We will also run your program with other input to make sure it behaves according to the specifications given in this statement.

If at any time when you are running your program, and you are "stuck" (the program has fallen into an infinite loop, or is unresponsive) you can "bail out" and halt the execution of the program using `Ctrl + C` (i.e., hold Control and C simultaneously).

You should be sure to include **THE NAME OF THE AUTHOR OF THE PROGRAM** in a comment at the top of your source code file, for this and all other assignments in this course.

Your source code must use proper **style**, that is variables should be well named (name is not too short, not too long, and is meaningful), and bodies of loops, if's, etc.. should be properly indented. Refer to the coding style file for this class, published on Canvas under Coding Standards in a file named: `cs241_codingStandards2020.pdf`

3 What to submit:

Submit your C program in a file named: `lastName_InitialFirstName_DNA.c`

4 Grading rubric:

If your C program does not compile with the `-std=c99 -pedantic -Wall` options without errors or warnings the points given for the assignment will be zero. Otherwise the following rubric will be used:

+ 10 pt: Your C files follow the class coding style.

+ 60 pt: Your `lastName_InitialFirstName_DNA.c` file passes the `diff` test using the first three input files in this assignment.

+ 30 pt: Your `lastName_InitialFirstName_DNA.c` file passes `diff` tests with dnaInput_4 and other local files available to the graders.