cs201 – sum 2020

Lab 8 – (15 points)

For information on how to do this lab, see the page on "Optimization" in the week 8 module.

Submit your modified source code files and, optionally, your answers in a text file to the assignment folder for lab 8 by the due date.

Part A: Vowel Count (10 points)

Do the following steps. Put your answers either in a separate text file or in comments inside your modified vowel_count.c file.

1. Download the file vowel_count.c from the assignment folder and upload this to the Linux PCC server.
   This file contains a function called countAllVowels() which takes an array of strings that are of random size with random letters in them. It then counts all the vowels (a,e,i,o,u) that are contained in all strings. The createLines() creates an array of random sized strings and then fills each string with lower-case letters chosen at random.
2. Let's look at this program in gprof. Compile the program using the following line:

   gcc -Og -pg vowel_count.c -o prog

   then run it to generate a file called **gmon.out:**

   ./prog

   NOTE: it takes about 1 1/2 minutes to run so be patient.
   now run gprof and redirect this to a text file so you can capture the output:

   gprof prog > out.txt

   Answer the question: What function has the most of the run time spent in this program?
3. Next, we'll be using time command to test how fast the unmodified program runs. Use the command line:

   time ./prog

   Now run the program three times and record the user values; convert them to seconds. Now find the average of these three values.

4. There are some performance problems with this code. Note that although the string size is different for each string, each string doesn't change it's size while the inner loop is running. So start by moving the strlen() call

somewhere where it isn't being called for every letter in the array (just for every string being processed). Now, re-compile with the same line as above and run it using time command:

```
time ./prog
```

run the program three times and record the user time values; find the average of these three values.
Answer this question: By what factor did the speed go down? Use the formula (#3 average)/(#4 average). Write this in your answer sheet or in comments in your code file.

5. There is also one more unnecessary call to a function inside this loop. Try getting rid of this call altogether (it's only do a series of comparisons) and do the same thing as above. How much of factor decrease is this from the above modified (the value from #4)?

Requirements

- **You must not change any part of the code except the function countAllVowels().**
- **The number of times each for loop runs should not change.**
- **The vowel count must be correct.**

Part B: Memory Usage in For Loops (5 points)

Do the following steps. Put your answers either in a separate text file or in comments inside your modified addition.c file.

1. Download the file addition.c from the assignment folder. This file contains a function called addition() which takes an two dimensional array of random numbers. It then adds all numbers within this array. Note that createTable is there to just create the array; it has negligible effect on run time. Also note that in main, the addition() function is being called 1000 times; this is to get enough time to pass to see the difference between the original and the modified function.

2. Compile the program using the following line:

```
gcc -Og addition.c -o prog
```

then run it use time command:

```
time ./prog
```

NOTE: it takes about 1 minute to run so be patient.

Do this three times and record the user time value each time. Average the values you get and record this in your answer sheet or in comments in your addition.c file. Also, look at the nested loops in addition() function and write down the **stride** for accessing this array. See "optimization" page or textbook chapter 6.2.1. These show some examples of what stride is.

3. There are some performance problems with this code. Note that the stride is very large and could be greatly improved. Change the indexing so that the stride is much better improved. Now, re-compile with the same line as above and run it using time command three times and record the user time values; find the average.
   Answer this question: By what factor did the speed go down? Write this in your answer sheet or in comments in addition.c.

Requirements

1. **You must not change any part of the code except the function addition().**
2. **The number of times each for loop runs should not change.**
3. **The sum must be correct.**