CSCI 1120 (Low-Level Computing), Fall 2012

Homework 3

Credit: 20 points.

1 Reading

Be sure you have read the assigned readings for classes through 10/01.

2 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 1120 homework 3"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (10 points) Complete the sample sort program we wrote in class by filling in the `sort` function. (You can find it linked from the course "sample programs" page here[1].)

   It's completely up to you which sorting algorithm to implement, though I'm inclined to recommend that you just do one of the simple-but-slow ones (e.g., bubble sort or selection sort). If you feel ambitious, you could try quicksort or mergesort, though mergesort is apt to be more trouble since it requires a work array. Please say in comments at the start of the program which sorting algorithm you're implementing. Feel free to make other alterations to the program (e.g., adding more functions, though depending on your choice of algorithm you might want to just do everything in `sort`).

2. (10 points) On the sample programs page is a simple recursive function to compute an element of the Fibonacci sequence and a main program to test it. If you compile and run this program, you will observe that it is quite slow for all but fairly small inputs. The reason for that is fairly obvious if you think about how it works — the simple function does quite a lot of duplicate computation. One way to improve the performance of such a function is with a technique referred to as *memoization*, in which every time we compute a result we save it for possible reuse. Your mission for this problem is to take the program from the course "sample programs" page here[2]), add a second recursive function to compute an element of the Fibonacci sequence using memoization, and compare the performance of the two functions. Sample output for an input of 46:

   ```
   result (simple implementation):  2971215073, time 23.6019
   result (memoized implementation):  2971215073, time 9.53674e-07
   ```

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2012fall/SamplePrograms/Programs/sort-example.c`

[2] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2012fall/SamplePrograms/Programs/fibonacci.c`

(I'm not fussy about details as long as you print both the result of the computation and how long it took.)

For the function, a simple way to use memoization is to have an array for saving previously-computed results, with the n-th element of the Fibonacci sequence stored as the n-th element of the array, and a value of 0 meaning "has not been previously computed". This array probably should be passed to the function, but it could be a global variable. The program should do something reasonable if this array is not big enough, perhaps just rejecting any input that would overflow it.

For comparing performance, what is useful is a function that gets the time of day with sufficient precision to allow for meaningful measurements of elapsed time. Unfortunately the C standard library does not have anything that makes that easy, but I wrote a Linux-specific function that gets the time of day as a `double`, which you can use. To do that, first get a copy of the file `timer.h`[3] and put it in the directory with your code. Sample usage of the function (notice the `#include` line):

```
/* other #include lines */
#include "timer.h"

    /* other code */

    double start_time, end_time;
    start_time = get_time();
    /* computation to time */
    end_time = get_time();
    /* print difference between end_time and start_time */

    /* other code */
```

[3] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2012fall/SamplePrograms/Programs/timer.h`