CSCI 1120 (Low-Level Computing), Fall 2010

Homework 3

Credit: 20 points.

1 Reading

Be sure you have read, or at least skimmed, the readings for 9/20, linked from the "Lecture topics and assignments" page[1].

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 looked at in class by filling in the `sort` function. (You can find it linked from the course "sample programs" page here[2].)

   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). We will probably do at least one of the faster recursive ones (quicksort and mergesort) later in the semester. 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) Write a C function that takes a character string as input and prints the string and, for each possible character, how many times it occurs in the string. Print counts only for the characters actually present. Also write a short main program that calls your function a few times with inputs showing that it works. Sample output might look something like the following:

```
input:
hello world 1234 !@#$

count for character 32 ( ) is 3
count for character 33 (!) is 1
count for character 35 (#) is 1
```

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2009fall/HTML/schedule.html`

[2] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2010fall/SamplePrograms/Programs/sort.c`

```
count for character 36 ($) is 1
count for character 49 (1) is 1
count for character 50 (2) is 1
count for character 51 (3) is 1
count for character 52 (4) is 1
count for character 64 (@) is 1
count for character 100 (d) is 1
count for character 101 (e) is 1
count for character 104 (h) is 1
count for character 108 (l) is 3
count for character 111 (o) is 2
count for character 114 (r) is 1
count for character 119 (w) is 1

input:
Now is the time for all good persons to come to the aid of their party.

count for character 32 ( ) is 15
count for character 46 (.) is 1
count for character 78 (N) is 1
count for character 97 (a) is 3
count for character 99 (c) is 1
count for character 100 (d) is 2
count for character 101 (e) is 6
count for character 102 (f) is 2
count for character 103 (g) is 1
count for character 104 (h) is 3
count for character 105 (i) is 4
count for character 108 (l) is 2
count for character 109 (m) is 2
count for character 110 (n) is 1
count for character 111 (o) is 9
count for character 112 (p) is 2
count for character 114 (r) is 4
count for character 115 (s) is 3
count for character 116 (t) is 7
count for character 119 (w) is 1
count for character 121 (y) is 1
```

Hints:

- Characters are actually small integers, and can be treated as such. If `c` is an `int` representing a character, you can print its numeric value and its value as a character with a line such as the following:
  `printf("%d %c\n", c, (char) c);`
- Possible integer values for characters range from 0 through `CHAR_MAX`, where `CHAR_MAX` is a constant defined in `limits.h`.

- There are several ways to approach this problem. It is probably slightly more interesting (and more pedagogically useful) if you limit yourself to solutions that scan the input string only once.