CSC172 LAB 16

C POINTERS

1 Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Labs will typically have an even number of components. The two partners in a pair programming environment take turns at the keyboard. This paradigm facilitates code improvement through collaborative efforts, and exercises the programmers' cognitive ability to understand and discuss concepts fundamental to computer programming. The use of pair programming is optional in CSC172. It is not a requirement. You can learn more about the pair programming paradigm, its history, methods, practical benefits, philosophical underpinnings, and scientific validation at http://en.wikipedia.org/wiki/Pair_programming .

Every student must hand in their own work, but every student must list the name of their lab partner, if any, on all labs.

This lab has six parts. You and your partner(s) should switch off typing each part, as explained by your lab TA. As one person types the lab, the other should be watching over the code and offering suggestions. Each part should be in addition to the previous parts, so do not erase any previous work when you switch.

The textbook should present examples of the code necessary to complete this lab. However, collaboration is allowed. You and your lab partner may discuss the lab with other pairs in the lab. It is acceptable to write code on the white board for the benefit of other lab pairs, but you are not allowed to electronically copy and/or transfer files between groups.

2 A Simple Linked List

The goal of this lab is to gain familiarity with pointers in the C programming language.

1. Begin this lab by implementing a simple C program with a “main” function. Use the prototype for main that is used with command line arguments below. Include a “for” loop that prints out the command line arguments, one argument per line. You may refer to the code in Kernighan and Ritchie if you need help.

```
int main(int argc, char * argv[]) {
     // your code here
}
```

2. Have the second member of your pair implement the pointer version of “strcmp”

   ```
   /* srtcmp: return < 0 if s < t, 0 if s==t, > 0 if s > t */
   int mystrcmp(char *s, char *t)
   ```

3. Implement a pointer version of “strcat” (string concatenate). Recall that the “array” version of strcat is found in chapter 2 of your book. You need to implement the pointer version.

   ```
   void mystrcat(char *dest, char *source)
   ```

4. Implement a pointer version of “strcpy” (string copy)

   ```
   void mystrcpy(char *s, char *dest)
   ```

5. Add an “if” statement to the “for” loop in your main routine. Use the “if” statement to check each command line argument. If the command line argument is either your name or your partner's name change it to print out a “Hello, ” before your name. All other command line arguments should be printed out with no change, one argument per line. You should do this by using strcpy to copy the eight characters in “Hello, ” into a char array 30 or so characters long. Use strcmp to check the argv char arrays (strings). When you find one of your names, concatenate the string containing your name to the big array containing “Hello, ” , then print out the resulting string.

6. Declare an array of int at least 10 elements long. Fill in the array with the square of it's index using array syntax, `a[i] = i * i`. Print out the array using pointer syntax `*(a + i)`.

3 Hand In

Hand in the source code from this lab at the appropriate location on the BlackBoard system at my.rochester.edu. You should hand in a single zip file (compressed archive) containing your source code, README, and OUTPUT files, as described below.

1. A plain text file named README that includes your contact information, your partner's name, a brief explanation of the lab (A one paragraph synopsis. Include information identifying what class and lab number your files represent.), and one sentence explaining the contents of all the other files you hand in.

   Please also advise your TA on your programming environment (e.g. Cygwin on Windows, gcc on Mac) so he/she knows how your code should be compiled.

2. Source code files representing the work accomplished for this lab. All source code files should contain author and partner identification in the comments at the top of the file.

3. A plain text file named OUTPUT that includes author information at the beginning and shows the compile and run steps of your code. The best way to generate this file is to cut and paste from the command line.

4 Grading

Each section (1-6) accounts for 15% of the lab grade (total 90%).

The README file accounts for the remaining 10%.