CS 240 (Park) **Midterm 2** **July 24 (Thu.), 2014**

Remarks: Keep the answers compact, yet precise and to-the-point. Long-winded answers that do not address the key points are of limited value. Binary answers that give little indication of understanding are no good either. Time is not meant to be plentiful. Make sure not to get bogged down on a single problem.

PROBLEM 1 (36 pts)

(a) Suppose someone wrote the following C code:

```
int x = 0;
if (x != 0)
   printf("A");
   x = 1;
   if (x == 1) {
      printf("B");
      x == 2;
   }
else
   printf("C");
```

What is printed to stdout as a result of running this code? Does a C compiler care about indentations when translating code?

(b) The following snippet of code

int *x; *x = 5;

is likely to generate a segmentation fault upon running. Why is that so? Fix the problem by using malloc().

(c) What happens when a process running a C program calls fork()? What is meant when people say that fork() "returns twice"? Why is this feature important for writing concurrent server apps?

PROBLEM 2 (32 pts)

(a) Suppose x is of type unsigned int whose length is 32 bits. You are asked to write C code using bit manipulation operations (shifting, 1's complement, masking, ANDing) that prints out the most significant bit of x (i.e., bit position 31 assuming bit positions start at 0). First, explain in words the steps involved in carrying out this task. Second, write C code that accomplishes this task.

(b) Suppose we have the following code snippet

float B[3][3]; B[1][2] = 10.7;

Rewrite the assignment statement B[1][2] = 10.7 using pointer arithmetic. That is, do not use the square brackets in the 2-D array notation. Explain how you arrive at the answer.

PROBLEM 3 (32 pts)

(a) What is the difference between an iterative server and a concurrent server? Why is an iterative server design not suited for writing a shell? Sketch the overall C code structure of a concurrent server as discussed in class. Where in the code should waitpid() be inserted? Why is waitpid() beneficial when implementing a shell? For what type of user input entered at a shell prompt is using waitpid() a hinderance?

(b) Suppose you need to read in a million real numbers from a file mill.dat and store it into an array of type double. Instead of declaring an array of type double of size 1000000, Z[1000000], you are asked to use dynamic memory allocation using malloc(). Write C code, including all variable declarations, that accomplishes this task. Use fscanf() to read the data. Assume that after reading in a million doubles from the data file, the program terminates.

BONUS PROBLEM (10 pts)

Why is fflush() useful when debugging C programs?