CS 240 (Park) **Midterm** **July 10 (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 x is of type `int` and the following function call is invoked from main(): scanf("%d",&x). Why is the ampersand preceding x needed? What is likely to happen if the ampersand is omitted? Why is the ampersand not needed when calling printf()?

(b) What is the role of the C preprocessor and how does it interact with the C compiler? Using two C preprocessor primitives used in class/labs, #include and #define, explain what their impact is after the C preprocessor runs.

(c) Declaring a variable y of type float outside the scope of all functions makes it global and declaring it within the scope of some function, say, myfunc(), makes it local to that function. Why would we declare y to be global in one app, while in another app we might choose to make it local? If y is global, how must myfunc() be coded so that it can access y? If y is declared local to myfunc(), is there no way for some other function, say, otherfunc() to gain access to the value of y? Explain.

PROBLEM 2 (36 pts)

(a) Suppose we have the following snippet of code:

```
int x, *y; x=5; y=&x; *y=10;
```

What is the value of x after the last statement? Explain what is going on.

(b) Suppose we have the following main() function:

```
main() { float *r; *r=100; printf("%f",*r); }
```

with stdio.h included at the top. What is likely to happen when this program is compiled and run? Explain.

(c) Suppose we have the code snippet:

```
char s[3], *u; s[0]='h'; s[1]='i'; printf("%s", s); u=s; *u='g'; *(u+1)='o';
```

First, is there a bug in the code? Second, what are the values of s[0] and s[1] after the last statement? Describe the effects of the last three statements.

PROBLEM 3 (28 pts)

(a) Suppose Mira writes C programs that run on hardware where type `int` is of size 32 bits. In one of Mira's apps, she needs to inspect the last two bits (i.e., bit positions 30 and 31, assuming bit positions start at 0) of variable z which is of type `int`. Using the bit manipulation techniques discussed in class (shifting, 1's complement, masking, ANDing), write a snippet of code that allows Mira to find the value of the two most significant bits of z.

(b) Explain the meanings of the following four declarations:

```
double *a[10];
unsigned int **b;
float *myfunc(int, int *);
char (*yourfunc)(void);
```

BONUS PROBLEM (10 pts)

The `static` modifier is another way to define a global, however, from within the scope of a function. How can this feature be useful and how is it different from the default way of declaring global variables (i.e., outside the scope of all functions)?