CS 316 Exam 2

Spring, 2008

Name:

This is a take home exam. Each student should do their own work on all of these problems. Unlike a homework assignment, you should not discuss your ideas on these problems with other students in the class. You can either print out the PDF version of this exam and write your answers on the printout, or you can type your answers into the MS Word version of this exam, or you can do a combination of the two. This exam is due, either in my physical mail box in CLO 316 or in my email mail box, by Wednesday, May 7.

Problem 1: (10 pts.) What do the following three loops compute?

```
         int i = 0;
         while (a[i++] != 0);
```

and

```
         int i;
         for (i = 0; a[i] != 0; i++);
```

and

```
         for (int i = 0; a[i] != 0; i++);
```

Are the three loops equivalent? That is, can you always replace the first loop with either of the other two?

Problem 2: (10 pts.) Give an example in C in which an in-line function may be significantly faster that a functionally equivalent macro. Give another example in which a macro is likely to be faster. (Hint: Think about applicative versus normal-order evaluation of arguments.) (This is problem 6.29, page 302, from the textbook.)

Problem 3: (10 pts.) Explain in detail exactly what the C expression

```
i + j / k
```

means and what its type is when the variables have the following data types. Be sure to specify any implicit type conversions.

- (a) `int i, j, k;`

- (b) `int i, j; double k;`

- (c) `double i; int j, k;`

Problem 4: (10 pts.) **(a)** In C (or C++) explain the difference in meaning between the following two lines.

```
enum x { a, b };
enum { c, d } x;
```

In particular, what does each `x` represent?

(b) The following code will compile in C but not in C++. Explain why. (Hint: Look up the two kinds of type equivalence.)

```
int main()
{
   enum { a, b } x;
   enum { c, d } y;
   y = c;
   x = y;
   return 0;
}
```

Problem 5: (10 pts.) Suppose that we have the following two arrays declarations.

```
int x[] = {1, 2, 3, 4};
int y[] = {5, 6, 7, 8};
```

Part (a): The assignment `2[x] = 0` will not compile in Java but it will compile in C. What does the assignment mean in C? Explain why.

Part (b): The assignment `x=y` will not compile in C but it will compile in Java. What does the assignment mean in Java? Explain why.

Part (c): Can the declarations be fixed so that the assignment `x=y` will compile in C? If so, then what exactly will the assignment do?

Problem 6: (10 pts.) In an if-statement

```
        if (exp1) exp2; else exp3;
```

the types of the expressions `exp2` and `exp3` do not matter at all. On the other hand, in

```
        exp4 ? exp5: exp6
```

the types of `exp5` and `exp6` do matter. In what way do the types matter in the conditional expression? Explain the reason for this difference in the two kinds of conditionals.

Problem 7: (10 pts.) Suppose that in C we try to write a short-circuit version of an `and` function in the following way.

```
         int and (int a, int b)
         {  return a ? b : 0;
         }
```

Why won't this work?

Problem 8: (10 pts.) Explain in detail the differences between the following three C functions.

```
int functionA()
{  static int i;
   i++;
   return i;
}

int functionB()
{  static int i = 0;
   i++;
   return i;
}

int functionC()
{  static int i;
   i = 0;
   i++;
   return i;
}
```

Problem 9: (10 pts.) Write a function, using C syntax, that will have three different effects, depending on whether the arguments are passed by value, by reference, or by value/result. Give an example of a specific function call that will has three different effects depending on the parameter passing method. Explain the details of how your function works with each kind of parameter passing method.

Problem 10: (10 pts.) To what extent is the following function tail recursive? Explain how a compiler that can do tail-call-optimization might optimize this function. In particular, describe how the runtime stack might evolve for the function call `f(-117)`. (How many function calls are there? How many stack frames would there be?).

```
int f(int n)
{
   if (n == 0)         return 0;
   else if (n < 0)     return f(-n);
   else if (n >= 10)   return f(n-10);
   else if (n > 0)     return 1+f(n-1);
}
```