CS 6311 – Programming Languages I – Fall 2006
Quiz #5 (20 pts)
Monday, 9/25/6

NAME:

1. Global variables are one type of non-local variables, given another example of a variable that is a non-local variable? (2 pts)

   A data member of a class. It is non-local to the methods declared within the class.

2. Provide code that will create stack-dynamic variables. Explaining how the variables end up on the stack. (4 pts)

```
int myMethod(int x)
{
        int y = 5;
        return x + y;
}
```

   Another method can have a call to myMethod at that time the local variables of myMethod (x and y) are allocated a memory location on the stack, when the myMethod is exited the variables of myMethod are popped (freed) from the stack.

3. Describe a situation that creates a scope hole. (3 pts)

   A data member of a class and a variable name within a method of the same class having the same name. In this instance, when in the method, the data member of the class is shadowed. Most languages provided a mechanism to access the shadowed data member variable by using the scope resolution modifier, *this* for this particular situation.

4. Explain how dynamic scoping can make programs more unreliable. (4 pts)

   With the variables visible within a method dependent upon the call sequence that activated the method, different invocations of the program could have different variables within scope for a particular method. For example, code within an if-else structure may effect what methods are called when affecting what variables are visible. Given this behavior, it is very difficult to thoroughly test dynamically-scoped code as one small variation in the call sequence that was not tested could have cause the program to have unexpected behavior. For these types of programs it is very difficult to construct all the possible test cases to ensure the program is robust.

5. What is the value of a at Location 1 and Location 2 below. (4 pts)

```
void main(void)
{
    static int c = 5;
    int b = 5;
    int a = 0;
    a = myfunc(a + b + c);
    // Location 1 a equals ???            16
    c = c + 1;
    a = myfunc(a + b + c);
    // Location 2 a equals ???            37
}

int myfunc(int a)
{
    static int c = 1;
    c = c + 2;
    int b = c + a;
    return c + b;
}
```

6. Explain why it is important to create test cases for boundary conditions. (3 pts)

   The boundaries are the most common places for programmer errors, e.g., not traversing all the way to the last element of the list, when comparing values using a less than instead of a less than or equal to, etc. These type of errors occur easily and often and therefore test cases need to be constructed to test the places where programming errors are most likely to occur.