CS 6311 – Programming Languages I – Fall 2006
Quiz #4 (20 pts)
Monday, 9/18/6

NAME:

1. True or False. Strong-type checking can be implemented in a dynamically-typed language. (2 pts)

   False. For a strongly-typed language type checking is enforced at compile and run-time.

2. Give an example language that uses static type binding and write a line of code in that language that statically declares a type. (2 pts)

   C#
   int myInteger;

3. Given the following lines of C# code. Explain why a type cast is not needed for the third line of code, but one is needed for the fourth line of code. (3 pts)

```
int myInt = 3;
double myDouble = 4;
double myVal = myInt + myDouble;
int anInt = myInt + (int)myDouble;
```

   For the third line of code the conversion of an int to a double needed for the myInt is a widening conversion so it is down automatically as there will be no lost of precision. However, for the fourth line of code the conversion of myDouble to an int is a narrowing conversion and this will not be done automatically because there could be a loss of precision with a narrowing conversion. Therefore, for a narrowing conversion an explicit cast forcing the conversion is needed.

4. Explain how a variable name provides a level of abstraction when programming. (3 pts)

   One attribute of a variable name is its address of where it is stored in memory, however when using the variable name allows you to access data in memory without knowing the actual memory address. Therefore, a variable name eliminates the need for absolute addressing and allows the programmer to store and retrieve data in memory without knowing the details about memory addresses.

5. C# allows two different type of constant declarations: const and readonly.

   a. Which attribute allows for dynamic binding of values? (1 pt)

      readonly

   b. The terms "dynamic binding of values" and constant seem to be contradictory. Explain what is meant by "dynamic binding of values" for a constant and how the constant value is enforced in C#. (4 pts)

      Dynamic binding of values for a constant allows an expression with a variable to be assigned to a constant at run-time, e.g. readonly int NUM = 3 * value; This allows for different invocations of the program to have different values assigned to the constant without having to manually modify and rebuild the code. C# enforces the constant part by forcing readonly constants to be fields of a class and the value assigned to the constant can only be assigned at declaration or in the constructor for the class and once it is assigned it cannot be changed.

6. It is possible to write two-lines of code to determine if a language is case sensitive or not, e.g. in Python the following two lines of code can be used to determine if Python is case-sensitive:

```
x = 2
print X
```

   However, in this instance you would have to understand the following run-time error message: NameError: name 'X' is not defined to determine this. To help the novice programmer, write a Python function called isCaseSensitive that returns a string indicating whether Python is case-sensitive or not.

   Hint: The function needs to create some variables and then use those variables to determine if Python is case-sensitive and return the appropriate string. For example, a decision construct may be useful where one branch would return Python is case-sensitive and the other branch: Python is NOT case sensitive. (5 pts)

```
def  isCaseSensitive():
     a = 3
     A = 4

     if a = = 4:
        return "Python is not case-sensitive"
     else:
        return "Python is case-sensitive"
```