Login name _____   **Quiz 2**   Name _____
                                   CSE 131B
Signature _____   **Spring 2007**   Student ID _____

1. Static vs. Dynamic Scoping. Consider the following C-like program:

```
int x = 9;

int f () { return x; }

int g () { int x = 2; return f(); }

int main()
{
  int x = input();      /* input() reads input from the user. */
  if ( x > 4 )
    output( g() );      /* output() prints out a value. */
  else
    output( f() );
}
```

Fill in the table of what value gets printed based on the user input and whether the language uses static or dynamic scoping.

Input	Static scoping	Dynamic scoping
4		
6		

2. Good Stuff

Given the following code fragment:

```
int x = 420;
float y = 4.20;
int *ptr1 = &x;
int *ptr2 = ptr1;
```

for each statement indicate whether it will cause a compile error or not? If a statement will cause an error, WHY? Treat each statement individually as if it was the only statement following the definitions above.

A) Compile Error
B) No Error

```
*&ptr2 = ptr1;          _____                 ++*(float *)&x;            _____


&*ptr2 = ptr1;          _____                 (float *)ptr2 = &y;        _____


ptr1 = *&ptr2;          _____                 ptr2 = ++&x;               _____


ptr1 = &*ptr2;          _____                 ptr2 = &++x;               _____
```

3. Check #5. Consider the following error messages and programs as defined in the starterCode and Project I:

```
E1) public static final String error5v_Call  =
       "Argument of type %T not equivalent to REF parameter %S, of type %T.";

E2) public static final String error5a_Call  =
       "Argument of type %T not assignable to value parameter %S, of type %T.";

E3) public static final String error5n_Call  =
       "Number of arguments (%D) differs from number of parameters (%D).";

E4) public static final String error5c_Call  =
       "Non-addressable argument passed to REF parameter %S (type %T).";
```

```
FUNCTION foo (a : FLOAT) : BOOLEAN;
BEGIN
    RETURN TRUE;
END foo;

VAR a : BOOLEAN;
VAR b : FLOAT;
VAR c : INTEGER;

BEGIN
    foo( (* Pass a var here *) );      List all variables which can be passed as an argument to foo()
    RETURN 0;                          without causing a compile error? _____
END.
```

Which error message (E1, E2, E3, E4) should be displayed if you tried to pass any of the variables that might cause a compile error? If no vars would cause an error, state "No Error". _____

```
FUNCTION foo (REF a : FLOAT) : BOOLEAN;
BEGIN
    RETURN TRUE;
END foo;

VAR a : BOOLEAN;
VAR b : FLOAT;
VAR c : INTEGER;

BEGIN
    foo( (* Pass a var here *) );      List all variables which can be passed as an argument to foo()
    RETURN 0;                          without causing a compile error? _____
END.
```

Which error message (E1, E2, E3, E4) should be displayed if you tried to pass any of the variables that might cause a compile error? If no vars would cause an error, state "No Error". _____

```
FUNCTION foo (REF a : FLOAT) : BOOLEAN;
BEGIN
    RETURN TRUE;
END foo;

VAR a : INTEGER;

BEGIN
    foo( a + 2.5 );          Which error message (E1, E2, E3, E4) should be displayed? If not an error,
    RETURN 0;                state "No Error". _____
 END.
```