Login name _____ **Quiz 2** **Name _____**

CSE 131B

Signature _____ **Spring 2005** **Student ID _____**

1. Project I Semantic Type Checking. Given the following Oberon code fragment:

```
VAR a : REAL;
VAR b : INTEGER;
VAR c : BOOLEAN;

PROCEDURE P ( x : INTEGER; VAR y : REAL );
BEGIN (* Procedure body not important. *) END P;

BEGIN
  (* Your example code would go in here. *)
END.
```

Give an example of an assignability error.

Give an example of an addressability error.

Give an example of a type equivalence error.

Give an example of a type inference rule the compiler will perform.

Regarding type checking, reference (VAR) parameters require the actual arguments to be

_____ and _____ to the formal parameter types while value parameters

require the actual arguments to be _____ to the corresponding formal parameter types.

2. Suppose a compiler can perform more elaborate context sensitive analysis than most commonly available compilers (like standard C/C++/Java compilers) and use the return type of a subroutine to help add additional context to determine which overloaded subroutine to bind/match. Consider the following C++ example:

```
      // Some variable to use/return in the functions below
      int a = 95;
      struct something { int z } b = { 95 };

      // Some overloaded functions
/* A */  int *foo( int x ) { return &a; }
/* B */  struct something foo( long x ) { return b; }
/* C */  int foo( double x ) { return a; }

      int
      main( void )
      {
        std::cout << ( 6.1 + foo( 6 ) ) << std::endl;

        return 0;
      }
```

You can imagine something similar written in Java or Oberon. The point is this (or an equivalent Oberon) program will not compile. Now assume the compiler can use the return type of a subroutine along with the argument type(s) to help find an appropriate subroutine based on the surrounding context. Using type coercions and this newfound type inference rule, which overloaded method (A, B, or C) would the compiler bind to successfully compile this program?

_____

Why?

3. Given the following declarations:

```
      int b[10] = { 0 };
      int *p = b;
```

Write 2 different statements (different ways) to assign `p` to point to the second array element in the array `b`.

1)

2)

Write 2 different statements (different ways) using `p` only (and not `b`) to assign the value 95 to the third array element in the array `b`.

1)

2)