Login name _____ **Quiz 4** **Name** _____

CSE 131

Signature _____ **Winter 2013** **Student ID** _____

1. What is the output of the following Reduced-C program:

```
int c = 5;

function : int foo( int x, int & y )
{
  int z = y;

  --x;
  ++y;
  --z;

  cout << c << endl;
  cout << x << endl;
  cout << y << endl;
  cout << z << endl;

  return y++;
}

function : int main( )
{
  int a = 9;
  int b = 2;

  a = foo( b, c );

  cout << a << endl;
  cout << b << endl;
  cout << c << endl;

  return 0;
}
```

Parameter passing mode for **x** _____

Parameter passing mode for **y** _____

Output

Fill in the blanks below to simulate the above in C. Basically what is really happening under the code.

```
int
foo( _______________ x, _______________ y )
{
  int z = _______________;

  -- _____________;

  ++ _____________;

  --z;

  printf( "%d\n", _____________ );

  printf( "%d\n", _____________ );

  printf( "%d\n", _____________ );

  printf( "%d\n", z );

  return ____________________;
}
```

```
int c = 5;

int
main( )
{
  int a = 9;
  int b = 2;

  a = foo( _____________, _____________ );

  printf( "%d\n", a );
  printf( "%d\n", b );
  printf( "%d\n", c );

  return 0;
}
```

Parameter passing mode for **x** _____

Parameter passing mode for **y** _____

(over)

2. Given the following variable definitions

```
int a = -99;         // initialized global variable
static int b;        // uninitialized static variable
```

write the SPARC assembly code which should be generated to properly allocate space for each along with their initial values and alignment and to ensure proper access/visibility to these variables if another file is linked to this code's object file.

```
        .section    _______________

        .align      ________

a:      ___________ -99

        ___________ a

        .section    _______________

        .align      ________

b:      ___________ ________
```

Assume local int * variables a and b are allocated space in a function's stack frame at memory locations

```
int * a   %fp-4
int * b   %fp-8
```

Complete the SPARC assembly instructions for the line

```
        b = a++;
```

that a Reduced-C compiler from this quarter might emit.

You can assume all the initializations of the local variables have been performed. Just emit the code to perform the expression on the right side of the assignment statement and assign the result into b.

We will need to use a temporary or two on the stack, so we will use location %fp-12 for tmp1 and %fp-16 for tmp2.

Follow the basic ld/ld/compute/st model.

```
ld      [________________], %o0

_______ %o0, [%fp - 12]           ! tmp1 = a



ld      [________________], %o0

_______ _________, %o1

add     %o0, %o1, %o0

_______ %o0, [________________]   ! tmp2 = a + ?



_______ [________________], %o0

_______ %o0, [________________]   ! a = tmp2



_______ [________________], %o0

st      %o0, [________________]   ! b = tmp1
```