1. Given the following Reduced-C code fragment:

```
Reduced-C

function : int foo( int & i, int j )
{
    return i + j;
}
```

Write the equivalent translated SPARC Assembly language statements emitted for function foo().

2. Perform step-wise peephole optimization on the following window of pseudo three-address instructions (max of three operands are allowed in an instruction – up to two source and one destination):

```
... other instructions ...

r3 = r2 * 32          (instruction eliminated)
r0 = r3
r0 = r3
r3 = 13 + 5
r3 = 13 + 5
r3 = 13 + 5

... other instructions ...
```

step 1          step 2          step 3
3. State which calling convention / parameter passing mode is being used and what gets printed:

Reduced-C

```c
int a, b, c;

function : int foo1 ( int d, int & e )
{
    d = 5;      Parameter passing mode for d _________________
    e = 14;      Parameter passing mode for e _________________
    return d + e + c;
}

function : int main()
{
    a = 21;
    b = 46;
    c = foo1( a, b );    Output:
    cout << "a = " << a << endl;    a = ______
    cout << "b = " << b << endl;    b = ______
    cout << "c = " << c << endl;    c = ______
    return 0;
}
```

Fill in the blanks of the equivalent C program to simulate the above Reduced-C parameter passing modes (that exposes what the compiler is actually doing to implement these parameter passing modes):

```c
int a, b, c;

int foo1( _____________ d, _____________ e )
{
    __________ = 5;
    __________ = 14;
    return ____________;
}

int main( void )
{
    a = 21;
    b = 46;
    c = foo1( _____________ , _____________ );
    printf( "a = %d\n", a );
    printf( "b = %d\n", b );
    printf( "c = %d\n", c );

    return 0;
}
```

What question would you like to see on the Final?