1. Given the following SPARC assembly code fragment that might be emitted in code gen for the statement

\[ x = a \times 256 - b; \]

```
ld [%fp -4], %o0 ! variable a allocated at %fp-4
set 256, %o1  ! const value 256; could have used mov instead of set
call .mul     ! multiply
nop            ! delay slot
!
st %o0, [%fp -20] ! store result of MulOp into temp at %fp-20
! End of Expr7 MulOp production

ld [%fp -20], %o0 ! ExprSTO with value stored at %fp-20
ld [%fp -8], %o1 ! variable b allocated at %fp-8
sub %o0, %o1, %o0 ! subtract
st %o0, [%fp -24] ! store result of AddOp into temp at %fp-24
! End of Expr6 AddOp production

ld [%fp -24], %o0 ! ExprSTO with value stored at %fp-24
st %o0, [%fp -12] ! variable x allocated at %fp-12
! End of AssignStmt production
```

Use peephole code improvement techniques discussed in class to transform the above code into a more efficient set of instructions with the same overall side effect of the value of the rhs expression being assigned to \( x \). Different types of code improvements may be worth more than others. Comments may be helpful for the grader.

Name the type(s) of code transformation / improvement you used. List all you used.
2. What gets printed in the following program? If a value is unknown/undefined or otherwise cannot be determined by the code given, put a question mark ("?") for that output. Hint: Draw stack frames!

**Reduced-C**

```c
int a = 23;
int b = 34;

function : void fubar( int * x, int & y, int * p, int & q )
{
    ++*x;
    ++y;
    ++*p;
    ++q;
}

function : void foo1( int & c, int * d )
{
    ++c;
    **++d;
    cout << a << endl; ______
    cout << b << endl; ______
    cout << c << endl; ______
    cout << *d << endl; ______
    fubar( &c, c, d, *d );
    cout << a << endl; ______
    cout << b << endl; ______
    cout << c << endl; ______
    cout << *d << endl; ______
}

function : int main()
{
    foo1( a, &b );
    cout << a << endl; ______
    cout << b << endl; ______
    return 0;
}
```

What question would you like to see on the Final?