ECE 468 & 573

Problem Set 3: Common sub-expression elimination and local register allocation

For the following problems, consider the following piece of three-address code:

```
1.  A = 7;
2.  B = A + 2;
3.  C = A + B;
4.  D = C + B;
5.  B = C + B;
6.  A = A + B;
7.  E = C + D;
8.  F = C + D;
9.  G = A + B;
10. H = E + F;
```

1. Show the result of performing Common Subexpression Elimination (CSE) on the above code.

2. Suppose E and C were aliased. How would that change the results of CSE?

3. In class, we discussed how aliasing might reduce the number of common subexpressions that we can eliminate. How might aliasing *increase* the amount of redundancy in the code. (hint: consider what would happen if B and D were aliased).

4. For each instruction, show which variables are live *immediately after the instruction*.

5. How many registers would be needed to perform register allocation with no spilling?

6. Top down register allocation is inefficient for the above code, as there are some variables that could safely be assigned to the same register. What are they?

7. Perform bottom-up register allocation on the code for a machine with three registers. Show what code would be generated for each 3AC instruction. When choosing registers to allocate, always allocate the lowest-numbered register available. When choosing registers to spill, choose the register holding a value that will be used farthest in the future (in case of a tie, choose the lowest-numbered register).

8. Draw the interference graph for the code.

9. (ECE 573 only) Perform register allocation via graph coloring for the code. If you need to spill, use the code-rewriting approach described in the notes.

Repeat steps 4, 7 and 8 for the following code (assume registers can hold either temporaries or variables):

```
1.  T1 = A + B;
2.  T2 = C + D;
3.  T3 = T1 - T2;
4.  T4 = C + T3;
5.  T5 = D + T3;
6.  T6 = T1 + T5;
7.  T7 = D + T6;
8.  B  = B + T7;
9.  T9 = A + T8;
10. A  = B + T8;
```