Compiler Fall 2011
PRACTICE Final Exam

This is a full length practice final exam. If you want to take it at exam pace, give yourself 75 minutes to take the entire test. Just like the real exam, each question has a point value. There are 75 points in the exam, so that you can pace yourself to average 1 point per minute (some parts will be faster, some slower).

Questions:

1. Types (25 points)
2. Frame Layout (10 points)
3. Translation to IR (10 points)
4. Register Allocation (15 points)
5. Dataflow Analysis/Optimization (15 points)
6. Domination (10 points)
7. Garbage Collection (15 points)

Question 1: Types [25 pts]

1. Show the typing derivation for the Tiger statement `x := f(r.a) + 3`. You may assume that your initial environment (Γ_0) has the following mappings (in addition to the base Tiger environment):
   $\Gamma_0(\text{x}) = \text{int}$
   $\Gamma_0(\text{a}) = \text{int}$
   $\Gamma_0(\text{r}) = \text{Record(a:string, b:int)}$
   $\Gamma_0(\text{f}) = \text{string} \rightarrow \text{int}$

2. Fill in the correct premises forthe sub-typing rule for function types. Then briefly explain why it is correct:

$$\frac{}{S_1 \to S_2 \sqsubseteq T_1 \to T_2}$$

3. Infer the type of the following ML function (show your work):

```
fun f (w,x) = case x of
                [] => []
              | y::z => w(y)::f(w,z)
```

4. What goes wrong if you attempt type inference on `fun f(x)= f`?

Question 2: Frame Layout [10 pts]

- Explain the concept of a static link. Why is it needed?

- Explain the difference between the stack pointer and the frame pointer. One of them can be omitted in certain circumstances. Identify which one, and explain when it is not needed, and what benefits are obtained from omitting it.

Question 3: Translation to IR [10 pts]

Translate the following bits of Tiger into IR (you can either draw the IR tree or write it out as SML constructors). For each case you should assume the following variable locations (all InFrame variables are in your own frame. You can refer to the frame pointer as simply FP). Note: you do not need to include bounds checks for array accesses:

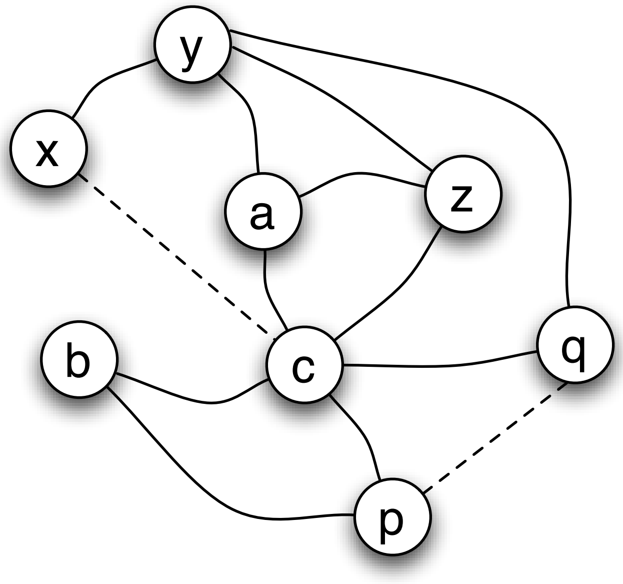
Variable	Location
x	InReg t1
y	InReg t2
z	InFrame -4
a	InReg t3

- y := 1 + x

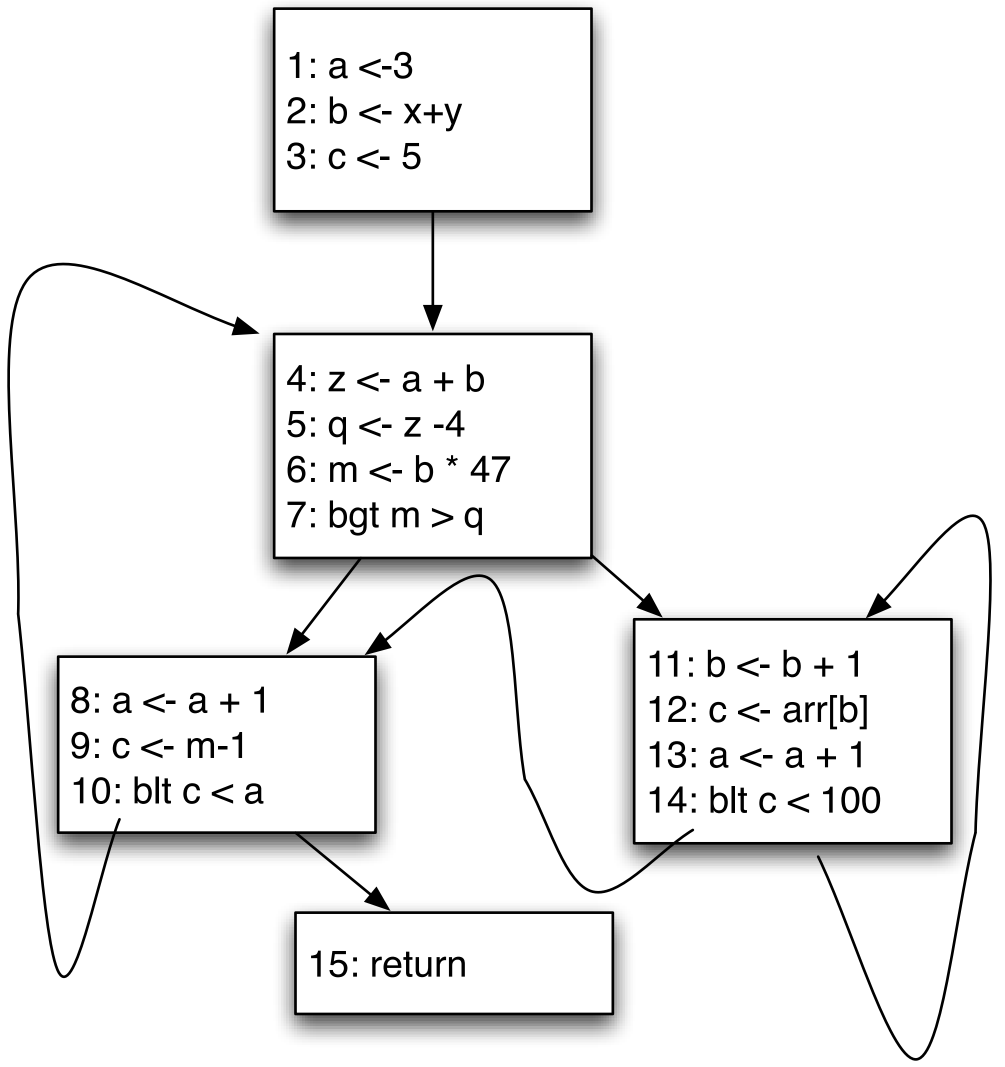
- a[y] := f(z)

Question 4: Register Allocation [15 pts]

Perform register allocation for a 3 register machine on the following interference graph (dashed lines indicate move relationships, solid lines indicate interference). You should coallesce moves whenever it is safe to do so according to either heuristic we learned. Show your work (you do not need to redraw the graph for each step, but you should list the order in which you simplify/coallesce/freeze nodes)



Question 5: Dataflow Analysis/Optimization [15 pts]



1. Using the above program fragment, which definitions reach the following uses (you can identify them by their instruction number):
   - The use of `a` in instruction 4.
   - The use of `a` in instruction 8.

- The use of `b` in instruction 6.

2. For the same program fragment, indicate whether each of the following expressions is "very busy" (write Y or N) after each block (after the last instruction in that block, numbered down the left side of the table):

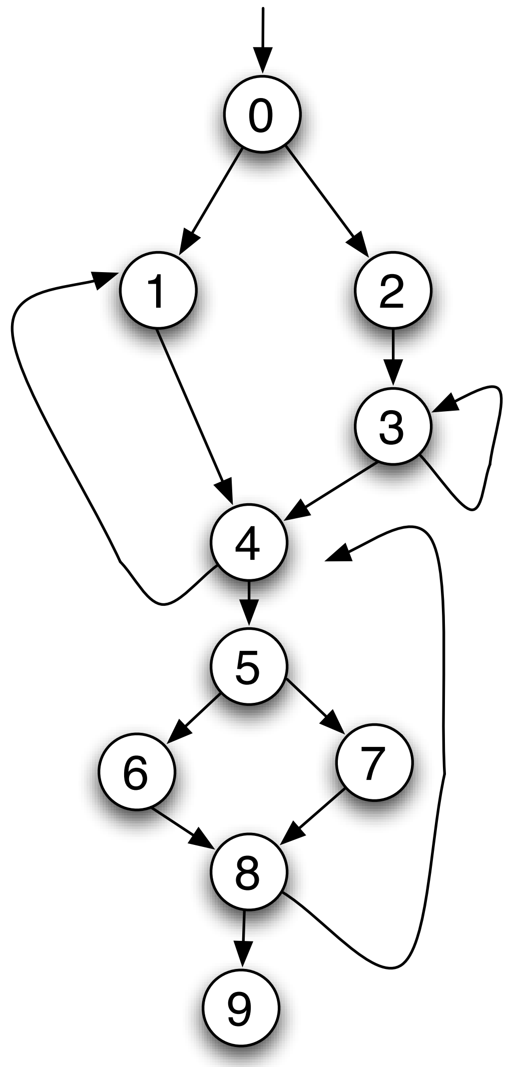
	a + 1	m - 1	a + b	b * 47	x + y	b+1	arr[b]
3							
7							
10							
14							
15							

3. For each of the following, indicate the appropriate data flow analysis (you can just write one of RD, LV, VBE, or AE on each line):

   - Common sub-expression elimination
   - Forward flow/union
   - Def/Use Web Formation
   - Backwards flow/intersection

Question 6: Domination [10 pts]

Consider the following control flow graph:



- Label each node in the graph with its dominator set.
- Identify the loops in the graph by their backedges.

- This control flow graph has a part that looks like a loop to a naive definition, but is not a loop for the definition required for many optimizations. Identify this false loop and briefly explain why it is not a loop.

- Draw the immediate dominator tree for this graph.

Question 7: Garbage Collection [15 pts]

- List three advantages of garbage collection. You may list advantages particular to one specific GC algorithm, but please specify which algorithm if you do so.

- Briefly describe the performance/space tradeoffs between the three algorithms we discussed: reference counting, mark and sweep, and stop and copy.

- The garbage collector needs to know which fields in an object are pointers, and which are not. Briefly describe two ways it might do this.