1. Register Allocation

For the following control flow graph, perform register allocation. Draw the interference graph and the assignment of registers (or colours) to variables.
2. **Software Pipelining**

Consider a loop with the following precedence constraints between its instructions, running on a machine with two resources.

![Diagram](image)

a. If we use only basic-block scheduling techniques (not using loop unrolling or pipelining), what is the best throughput achievable for this loop?

b. What is the lower bound on the initiation interval as computed from the resource and precedence constraints of this loop? Show your working.

c. Will the software pipelining algorithm described in class find a schedule meeting the lower bound?
   i. If yes, please show the schedule of an iteration of the pipelined loop and its modulo resource reservation table.
   ii. If no, give a one-line reason. Is it possible to find a schedule meeting the lower bound (i.e. not using the algorithm)? If yes, please show the schedule of an iteration of the pipelined loop and its modulo resource reservation table.
3. Global Instruction Scheduling:

SLOWMO is a statically scheduled machine that can only issue one operation every clock. Its memory load operation (denoted by the * dereference operator) has a latency of three clocks; all other operations have a latency of one clock. Consider the following locally scheduled program:

Assume that 'r' is the only live variable at the end of the program. Each branch in the flow graph is labeled with the likelihood that it is taken dynamically.

a. Assuming that $p = 0.01$, what is the best globally scheduled code that you can generate for the above program? You may apply any of the code motions discussed in class and introduce extra nops to fit the schedule. What is the expected execution time of the scheduled program?

b. Repeat part (a) for $p = 0.99$.

c. Find the expected execution time of the scheduled program in part (a) and part (b) under $p = 0.40$. Are they better than the original program?
4. Pointer Analysis:

Consider the following snippet of Java:

```java
public class Bazinga {
    public Bazinga a;
    public Bazinga b;
    public static void main(String [] args) {
        Bazinga x = new Bazinga(); // h1
        Bazinga y = new Bazinga(); // h2
        Bazinga z = new Bazinga(); // h3
        bar(x, y);
        bar(z, z);
        z.a = x;
    }
    public static void bar(Bazinga z1, Bazinga z2) {
        if (z1 == z2) {
            return;
        }
        z1.a = z2;
        baz(z1);
        baz(z2);
    }
    public static void baz(Bazinga z) {
        if (z.b != z)
            z.b = z;
        else
            baz(z.b);
    }
}
```

For reference, here are the Datalog rules for a context-insensitive analysis:

```datalog
vP(v,h) :- vP0(v,h).
vP(v1,h) :- assign(v1,v2), vP(v2,h).
hP(h1,f,h2) :- store(v1,f,v2), vP(v1,h1), vP(v2,h2).
vP(v2,h2) :- load(v1,f,v2), vP(v1,h1), hP(h1,f,h2).
```

a) What are the hP tuples inferred from this code in a **context-insensitive** analysis? To get you started, we give you one of the tuples: hP(h1,a,h2)
For reference, here are the Datalog rules for a context-sensitive analysis:

\[
\begin{align*}
\text{vP}(\_\text{v}, h) & :\text{vP}(\text{v}, h). \\
\text{vP}(c_1\text{v}, h) & :\text{assignc}(c_1\text{v}, \text{v}_1, c_2\text{v}, \text{v}_2). \text{vP}(c_2\text{v}, h). \\
\text{hP}(h_1f, h_2) & :\text{store}(\text{v}_1f, \text{v}_2), \text{vP}(c, \text{v}_1, h_1), \text{vP}(c, \text{v}_2, h_2). \\
\text{vP}(c, \text{v}_2, h_2) & :\text{load}(\text{v}_1f, \text{v}_2), \text{vP}(c, \text{v}_1, h_1), \text{hP}(h_1f, h_2).
\end{align*}
\]

b) What are the hP tuples inferred from this code in a context-sensitive analysis?

c) How many copies of bar are made for a context-sensitive analysis?

d) How many copies of baz are made for a context-sensitive analysis?