This is an open-book, open-notes, open-laptop, closed-network exam. Please do not post anything on Piazza until the solutions are put up on the class website.

You have 1 hour 20 minutes to work on this exam. The examination has 5 problems worth 70 points. Please budget your time accordingly. Write your answers in the space provided on the exam. If you use additional scratch paper, please turn that in as well.

Your Name: __________________________ SUNet ID: __________________________

The following is a statement of the Stanford University Honor Code:

1. The Honor Code is an undertaking of the students, individually and collectively:

   (a) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

   (b) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

2. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

3. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

Signature: __________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>15</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>25</td>
<td>70</td>
</tr>
</tbody>
</table>
Problem 1. True or False? Briefly justify your answer in 1 to 5 lines [15 Points]

1. A monotone forward data flow analysis must set OUT[Entry] to the “top” of the semilattice, and a backward flow must set IN[Exit] to the “bottom”.

2. Partial Redundancy Elimination can introduce speculative execution (execution of instructions that have no side effects but potentially would not be executed by the original program).

3. Two natural loops (arising from two back edges) can intersect without being completely nested one into the other.

4. Given a machine with \( n \) registers, without the spilling heuristic, if register coloring gets stuck, then rerunning the algorithm and picking nodes in a different order may find a coloring successfully.

5. Given a control flow graph, the set of retreating edges is unique.
Problem 2. Register Allocation [5 Points]
Consider the following interference graph:

On a machine with 3 registers, will the heuristic coloring algorithm always, sometimes, or never find a coloring (with spilling)?
Problem 3. Dominators [10 points]
Consider the following control flow graph:

- Draw the immediate dominator tree for the flow graph above.

- What are the natural loops in the above flow graph? Indicate both the set of blocks and the corresponding back edge.
• Is the flow graph reducible? Explain your answer.

**Problem 4.** Partial Redundancy Elimination [15 Points].
Show the result of running partial redundancy elimination. What’s the final optimized flow graph? You don’t need to show the intermediate steps.
Problem 5. Taint Analysis [25 points]

Given a string variable $x$, we say that $x$ is tainted if any portion of $x$ can potentially come from input from the user. Your goal is to design an analysis that can detect usage of variables that are potentially tainted. This is useful to detect common vulnerabilities such as SQL Injection, Cross Site Scripting and others.

Your language supports the following operations:

- $x = \text{const}$: set a variable to a constant
- $z = x + y$: concatenate the strings $x$ and $y$
- $x = \text{input}()$: read from the input
- $\text{use}(x)$: use the variable $x$

You can treat uninitialized variables as you prefer, but state your assumptions.

1. Design a dataflow analysis that can detect potentially tainted variables. Is it a forward or backward pass? What is your semi-lattice? What is the transfer function?

2. Is your framework monotone?

3. Is it distributive?

4. How do you warn for tainted variables?

5. Suppose we add another operation, $y = \text{sanitize}(x)$. This operation reads $x$ and transforms it into a string that is not tainted. What is the transfer function for this new instruction?
6. We now extend the language with pure function calls. These calls take arguments and return values, but cannot have side effects, and cannot modify local variables (e.g. through pointers). What is the new transfer function? You are not allowed to analyze across functions.

7. Finally, we generalize the language to arbitrary function calls, include those that can indirectly call `input`. How do you conservatively define the transfer function in this case? Again, assume that local variables cannot be modified (e.g. through pointers) by the function call, and you cannot see the code inside the called function.