CS243 Midterm Examination
Winter 2015-2016

February 11th, 2016
3:00 pm - 4:15 pm

This exam is open book/laptop. No internet access is allowed.
Power is not guaranteed for the laptop.

Duration: 75 minutes

Please do not post anything on Piazza until the solutions are put up on the class website.

Answer all questions on the exam paper itself.

Write your name here: _____

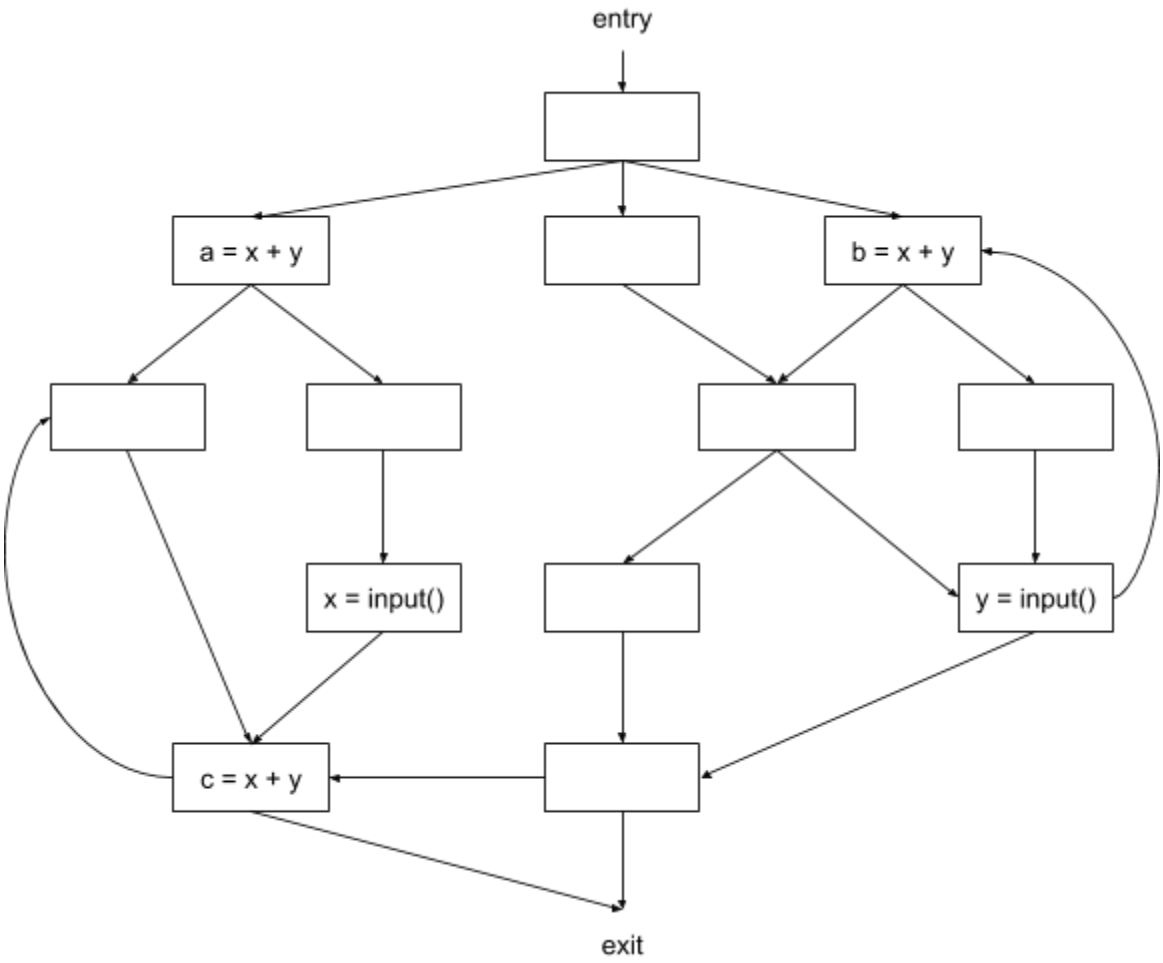
I acknowledge and accept the Stanford honor code.

(signed) _____

Question	Marks	Score
1	14	
2	12	
3	12	
4	22	
Total	60	

Q1) Partial Redundancy Elimination [14 marks]

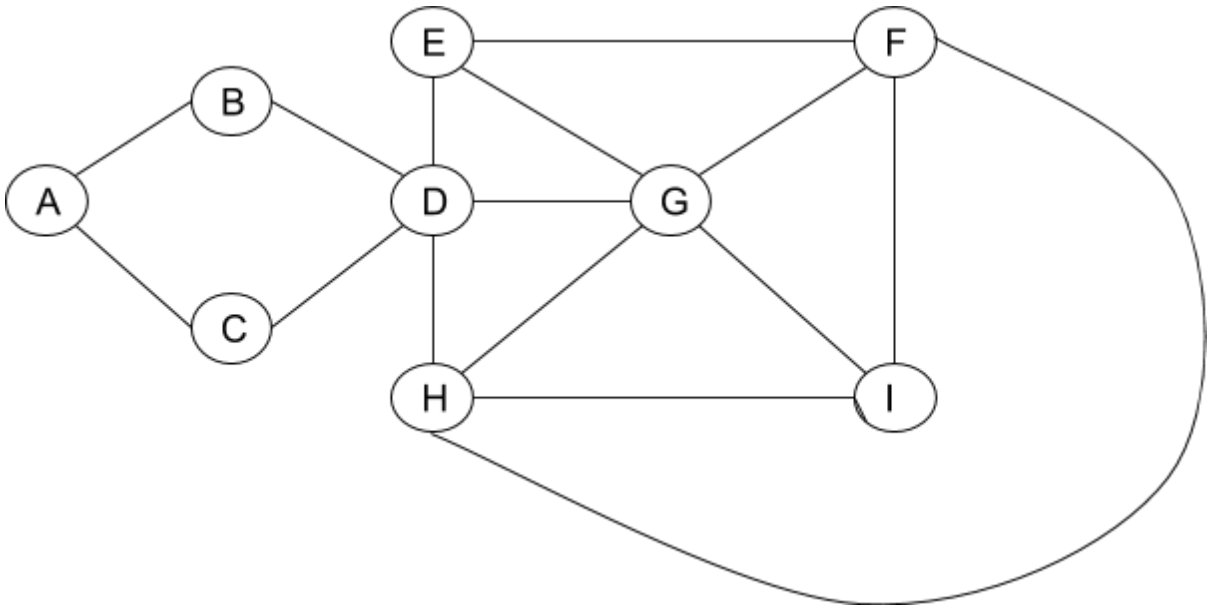
Show the result of running PRE. You don't need to show any intermediate steps, only the final optimised flow graph.



Q2) Register allocation [12 marks = 2 + 1 + 9]

a) True/False: In an interference graph, it is never possible to find a valid coloring for a node with degree = n, where n is the number of registers available. Provide a justification by way of a short proof or counterexample.

Consider the following register interference graph:

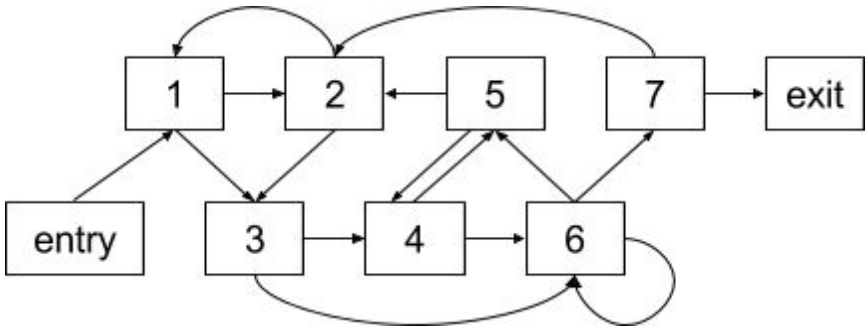


b) What is the minimum number of registers, R, required to avoid spilling for the above interference graph?

c) Show a register allocation that uses R number of registers for this graph. You can label the nodes in the graph with the register number assigned.

Q3) Dominators and Natural Loops [12 marks = 6 + 4 + 2]

Consider the following control flow graph:



a) Draw the dominator tree for the above graph

b) Identify all the back-edges and their corresponding natural loops

c) Is the above graph reducible? (Yes/No) Provide the reason in a single line.

Q4) Simplified Pointer Analysis [22 marks = 2 + 10 + 10]

Pointers are an extremely useful method for indirect access but are also the cause of many bugs when not handled carefully. Improper allocation or deallocation of pointers can lead to dangling pointers and memory leaks which can cause segmentation faults in run time.

Consider a simplified programming language with two kinds of statically typed variables: (1) integers, or (2) pointers to integers. For simplicity, we use p, q to represent pointer variables, and u, v, w to represent integer variables. It has the following statements:

ASSIGNMENT:	v = <constant>
COPY INT VARIABLES:	u = v
ADDITION:	w = u + v
COND BRANCH:	if (v) goto L
ALLOCATION:	p = new()
LOAD INT INDIRECTLY:	v = *p
STORE INT INDIRECTLY:	*p = v
DELETION:	delete(p)

Your task is to develop a static program analysis that will issue the following warnings:

- Type 1 Warning (Dangling pointer): Issue a warning on any statement that may dereference or delete an invalid pointer. A pointer is invalid if has not been allocated, or if it has been freed.
- Type II Warning (Memory Leak). Issue a warning on an allocation statement if the pointer MAY not have been deleted before the program exits.

For example for the following code:

```
L1: p = new()
L2: v = 2
L3: *p = v
L4: if(v) goto L6
L5: delete(p)
L6: v = *p
L7: if(v) goto L9
L8: p = new()
L9: delete(p)
```

a. Warning I should be issued for: {L6, L9}

b. Warning II should be issued for: {L1}

You can design **one or more** data flow analyses to answer the two parts of this problem. On the next page is a template that you need to fill out for each analysis.

State explicitly how you generate the warnings and the causes of the warnings.

Data Flow Analysis Template (Warning I)	
Property	**Answer**
Direction of Analysis (Forward/Backward)	
Meaning of the values in the semi-lattice	
Semi-lattice diagram (Mark the top and bottom clearly)	
Meet Operator	
Transfer function of a basic block	
Boundary condition (assignment to OUT[entry]/IN[exit])	
Interior Points (assignment to IN[B]/OUT[B])	
Is the framework monotone? (Yes/no: No explanation needed)	
Is the framework distributive? (Yes/no: No explanation needed)	
Will the algorithm converge? (Yes/no: No explanation needed)	

Data Flow Analysis Template (Warning II)	
Property	**Answer**
Direction of Analysis (Forward/Backward)	
Meaning of the values in the semi-lattice	
Semi-lattice diagram (Mark the top and bottom clearly)	
Meet Operator	
Transfer function of a basic block	
Boundary condition (assignment to OUT[entry]/IN[exit])	
Interior Points (assignment to IN[B]/OUT[B])	
Is the framework monotone? (Yes/no: No explanation needed)	
Is the framework distributive? (Yes/no: No explanation needed)	
Will the algorithm converge? (Yes/no: No explanation needed)	

<This page is intentionally left blank for spill-over>

<This page is intentionally left blank for spill-over>