Name_____ EE573 or EE468 _____

**** FORM A **** FORM A **** FORM A **** FORM A **** FORM A ****

Section I, Multiple choice questions

Mark all answers that you consider valid for explaining compiler technology to a less-skilled student. A valid answer should apply to the typical case, unless it has an attribute, such as "always" or "never".

If you think a question is ambiguous, state your assumptions.

(Each correct answer counts as one point)

1. Syllabus:
a- Except in rare circumstances I will receive a 0 grade on a missed exam.
b- Before the semester ends, I will need to verify that all project and exam grades are correct on the TA's record.
c- The last subproject is for EE573 students only.
d- Project deadlines are hard deadlines.
e- The final project step will have the most weight.

2. Basic Blocks:
a- Basic blocks have at least one branch instruction.
b- Basic blocks have a single entry and a single exit.
c- Local optimizations operate within basic blocks only.
d- Basic blocks have fewer than 1000 instructions.
e- Global optimizations are techniques that works interprocedurally.

3. Register Allocation:
a- The goal is to allocate as many registers as possible.
b- Reducing load and store operations is one goal of register allocation.
c- Optimal register allocation is of complexity $O(n^3)$.
d- Copying values to other registers in order to free a register is called spilling.
e- Deciding which register to free is among the most complex tasks of register allocation techniques.

4. Order of register allocation and instruction scheduling:
a- Register allocation should always be done before instruction scheduling.
b- Instruction scheduling should always be done before register allocation.
c- Doing register allocation first limits the amount of instruction reordering that can be done by instruction scheduling.
d- Register allocation and instruction scheduling are mutually dependent.
e- In practice, register allocation and instruction scheduling are implemented as two separate passes in order to limit the complexity of compilers.

5. Instruction Scheduling:

a- In some architectures, dependences between instructions are enforced by the processor.
b- For some architectures, dependences between instructions must be enforced by the compiler.
c- In out-of-order processors an instruction that consumes a certain value can be moved prior to the instruction producing this value.
d- Instruction scheduling typically reduces a code's execution time by a few percent.
e- In a processor, one or several instructions can be started in each cycle.

6. The following may be used to form instruction scheduling priorities:

a- latency on the instruction,
b- cumulative latency of the instruction,
c- number of predecessors on the instruction,
d- number of consumers of the instruction's result,
e- distance in number of instructions from the end of the basic block.

7. Program Optimization:

a- Most program optimizations involve more than a single basic block.
b- Unsafe optimizations can only be applied if allowed by the program's profile.
c- If the program's input data is known, the profitability of optimizations can be determined by the compiler.
d- The interdependence of optimizations can be resolved by running all optimizations twice.
e- All program optimizations need some form of program analysis.

8. Benefit from program optimization:

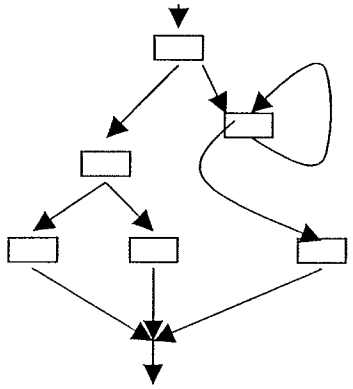
a- Sequential program optimization typically leads to a 2 to 4-fold speedup.
b- Replacing an algorithm with a better one can lead to orders of magnitude improvement.
c- Compilers only apply optimizations that are guaranteed to improve the performance.
d- There is little room for improving today's compiler optimization capabilities.
e- Optimizations that may have a negative effect are often implemented in combination with a compiler switch that lets the user turn them off.

9. Program analysis:

a- Control flow analysis finds predecessor-successor relationships among statements.
b- Dataflow analysis is a compiler technique for dataflow architectures.
c- Data-dependence analysis is used to find ordering constraints among instructions.
d- Alias analysis discovers data references that may access disjoint storage locations.
e- Program optimizations include a program analysis and a program transformation part. The analysis consumes more compile time.

10. Consider the following program and its control-flow graph.

```
d = 10
if (c==0) then
  if (a==1) then Print("it's a correct CFG")
  else  Print("no way!")
  endif
  d = d+10
else
  while (d>5) do
    Print("so much work...")
    d = d-1
  endwhile
  c=10
endif
```



a- The CFG correctly corresponds to this program:
b- (a) is false but would be true if the While loop were a For loop.
c- (a) is false but would be true if the statement "d=d+10" were deleted.
d- This CFG is acyclic as are all CFGs.
e- This program does not correspond to a valid CFG.

11. Interprocedural analysis:
a- In today's compilers, most optimizations work interprocedurally.
b- Interprocedural analysis makes available to subroutines information gathered about other subroutines.
c- Interprocedural analysis is always iterative.
d- Alias analysis must be done interprocedurally.
e- Strength reduction analysis may involve interprocedural analysis.

Section II, Solve the following problems on a separate sheet of paper. Write the answers in the provided space.
(Each subquestion counts as five points)

12. Consider the bottom-up register allocation algorithm for the following code:
(assume an architecture with 3-address instructions)

```
add A B C
add D E F
mul C F G
add B G H
```

Question1: How many registers will you need to avoid spilling? _____
Question2: Given an architecture with only two registers, how many additional loads and stores does the algorithm generate compared to (1) ?
Additional loads _____, additional stores _____.

13. Consider the tree code generation algorithm for the following code (same assumptions about architecture and machine code as in text book. There is only one available register):

d/e + a*(b+c)

Question 1: What is the register need? _____
Question 2: What is the sequence of cases (1 through 4) taken by the algorithm when traversing the tree? (don't indicate a case if the node is a leaf) __,__,__,__,__,__,__,__,. . .
Question 3: How many registers would conventional code generation take? _____

14. Advanced question (This is a bonus question for EE468 students)
Consider the list scheduling algorithm for the following code:
(Assume same latencies as on course slides)

```
load A  → r0
load B  →r1
mul r0 r1 →r0
load C  →r1
load D  →r2
div r1 r2 →r1
sub r0 r1 →r0
store r0  →E
```

Question 1: How many virtual registers do you need? _____
Question 2: What is the cumulative latency of the first instruction?_____
Question 3: How many cycles is the execution of the original code? _____
Question 4: How many cycles is the execution of the scheduled code? _____