CS3300: Final Exam: Nov 18 2019 (A)

Maximum marks = 40, Time: 3.00 hrs

Name: ______________________________ Roll: ______________

- Write your roll number on every sheet of the question paper and first page of the "rough" answer book.
- Every question is an MCQ for two marks (default) or one mark (explicitly mentioned).
- **Negative marking**. Each incorrect answer will lead to deduction of 1 mark (for a two marks question) or 0.5 mark (for a one mark question).
- You may do any rough work in the "rough" answer book; start from page 3. You may make any reasonable assumptions that you think are necessary; but state them in the first two pages of the rough book clearly (by mentioning the question number and any associated assumptions). Your "rough" work will NOT be evaluated.
- Total questions=25. Total marks=42.
- **Mark your answers on the question paper and return the same, along with your rough answer book**.

---

1. Consider the code shown in Question 1. The number of basic-blocks which have exactly three dominators is equal to

   (a) 0

   (b) 1

   (c) 2

   (d) 3

2. Consider the following grammar and SDT actions.

```
B0 ->  B1 0       {B0.twoOrMore1s = B1.twoOrMore1s;  B0.1ctr=0;}
     | B1 1       {B0.1ctr = B1.1ctr + 1; B0.twoOrMore1s++;}
     | 0          {B0.1ctr = 0; B0.twoOrMore1s=0;}
     | 1          {B0.1ctr = 1; B0.twoOrMore1s=0;}
```

   The goal of the SDT was to count the number of sequence of two or more 1s. For example, given a string 011011101101, `B.twoOrMore1s` should contain 3. Which of the following is true, for all inputs?

   (a) The SDT works as expected.

   (b) In the second SDT rule `B0.twoOrMore1s++;` needs to be replaced with `if (B0.1ctr==2) B0.twoOrMore1s++;`

   (c) In the second SDT rule `B0.twoOrMore1s++;` needs to be replaced with `if (B0.1ctr%2 == 0) B0.twoOrMore1s++;`

   (d) None of the above.

3. Consider the following grammar:

   S → A a A b | B b B a
   A → B | a
   B → A | b

   Which of the following statements is/are true about it?

   (a) The grammar is LL(1)

   (b) An equivalent grammar exists which is LL(1)

   (c) The grammar is not LR(1)

   (d) None of the above.

4. Consider the following piece of C code.

```
void foo(){
        x = a + b;
        y = a + b + c;
        L0: if x > y goto L1
                y ++;
        L1: goto L2;
        x = y + 3;
        L2: goto L0
}
```

Which of the following optimization(s) make(s) no difference to the above code?

(a) Common subexpression elimination

(b) Fuse jump statements

(c) Loop invariant code motion

(d) Copy propagation

5. Consider the following C code.

```
x = 0
L1: z = y + x;
    y = x + 1
    z = z + y
    x = y / 2
    if x < N goto L1
return z
```

Post liveness analysis, the OUT set for Line L1 is

(a) {x, y, z}

(b) {x, z}

(c) {x, y}

(d) {y, z}

6. Consider an IR that has the following three address instructions: binary statement (e.g., x = y op z), copy (e.g., x = y), copy positive constants (e.g., x = +1), unconditional branch (goto L), conditional branch (if (x) goto L).
Consider the following C code:

```
        if (y > z) { p = -q; }
        else { x --; }
```

The minimum number of IR instructions (don't count the labels) required to encode the above code is

(a) 5

(b) 7

(c) 9

(d) 11

7. Consider the below table listing three compiler phases and three typical error messages.

	Phase name		Error name
P.	Pre-processing / macro-processing	i.	Variable declaration not found.
Q.	Syntax Analysis	ii.	Semicolon missing
R.	Semantic Processing	iii.	Included header file not found

Match errors and the phase in compilation that throws the error:

(a) P - i, Q - iii, R - ii

(b) P - iii, Q - ii, R - i

(c) P - ii, Q - iii, R - i

(d) P - iii, Q - i, R - ii

8. Consider the following grammar for specifying boolean expressions and some semantic actions:

```
D =  int id;    {remember the type of id}
   | bool id;   {remember the type of id}
E =  E1 > E2    {check (E1.type == E2.type == integer); E.type = bool;}
   | E1 OR E2   {check (E1.type == E2.type == bool);    E.type = bool;}
   | E1 AND E2  {check (E1.type == E2.type == bool);    E.type = bool;}
   | !E1        {check (E1.type == bool);               E.type = bool;}
   | id         {                                       E.type = bool;}
P = D* E*       // Series of declarations, followed by series of expressions.
```

Here, `id` represents a variable, which can be of type integer or boolean. The goal of the SDT is to perform type checking. Which of the following holds true for the above SDT.

(a) The SDT does type checking as expected.

(b) The SDT correctly type checks expression containing only boolean variables (and operations thereof).

(c) The SDT correctly type checks expressions containing only integer variables (and operations thereof).

(d) None of the above.

9. Consider the following C code. Assume that `A`, `B` and `C` are floating point arrays.

```
for (int i=0;i<n;++i)
  for (int j=0;j<n;++j) {
          C[i,j]=0;
        for (int k=0;k<n;++k)
              C[i,j] += A[i,k] * B[k,j];
  }
```

The number of floating point multiplication, addition, loop-index increments, and comparison operations respectively, present in the code are:

(a) n^3, n^3, n^3, n^3

(b) $n^3, n^3, n^3 + n^2 + n, (n+1)^3 + (n+1)^2 + (n+1)$

(c) $n^3 + n^2 + n, n^3 + n^2 + n, n^3 + n^2 + n, n^3 + n^2 + n$

(d) n, n, n, n

10. Consider the following string
$3 + 2 + 8 * 4/6 * 7$.
Which of the following statements is true?

   (a) The height of the syntax tree is independent of the exact grammar used to parse the above string.
   (b) The leaves of the AST is dependent on the exact of grammar used to parse the above string.
   (c) The minimum number of syntax-tree nodes for the above string is 1.
   (d) The maximum number of syntax-tree nodes for the above string is 12.

11. Consider an IR that has only assignment statements and unconditional goto statements. Which of the following rules are enough to describe the computation of dominators information.

   (a) $Dom(n) = \cap_{x \in succ(n)} Dom(x)$
   (b) $Dom(n) = Dom(x)$, where x is the first element of the set $succ(n)$
   (c) $Dom(n) = Dom(x)$, where x is the first element of the set $pred(n)$
   (d) None of the above.

12. Which one of the following statements is true?

   (a) For every program a flow-sensitive analysis is more precise than flow-insensitive analysis.
   (b) For every program a context-sensitive analysis is more precise than context-insensitive analysis.
   (c) For every program an inter-procedural analysis is more efficient than intra-procedural analysis.
   (d) None of the above.

13. Consider an IR that has only assignment statements and unconditional goto statements. Which of the following rules are enough to describe the liveness analysis?

   (a) $Out(n) = In(x)$, where x is the first element of the set $succ(n)$
   $In(n) = Use(n) \cup (Out(n) - def(n))$
   (b) $Out(n) = \cup_{x \in pred(n)} In(x)$
   $In(n) = Use(n) \cup (Out(n) - def(n))$
   (c) $Out(n) = In(x)$, where x is the first element of the set $pred(n)$
   $In(n) = Use(n) \cup (Out(n) - def(n))$
   (d) None of the above.

14. Consider the following two instructions part of a serial program:

```
...           // Some code not shown
LD R1 [M1] // loads from the designated memory location M1 to register R1
ST [M1] R1 // stores to the designated memory location M1 from register R1
...           // Some code not shown
```

Which of the following statements holds true for an optimization to be semantics preserving.

   (a) Both the statements can be unconditionally removed.
   (b) Only the first statement can be unconditionally removed.
   (c) Only the second statement can be unconditionally removed.
   (d) None of the above.

15. Consider the following instructions of an intermediate representation. Here, v1, v2 etc represent variables and L1, L2 etc represent labels.

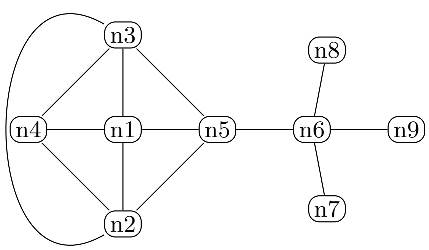
```
v1 = v2 op v3        // op is any binary op.
v1 = v2
v1 = C                // to copy arbitrary constant literal
goto L1
if v1 goto L1
L1: nop               // no-op instruction.
```

What is the minimum number of IR instructions required translate the following C code (assume that p, q, x, y and z are variables:

```
if (p+q > x+y) p++; else x = x + p + z;
```

(a) 10
(b) 11
(c) 12
(d) 13

16. Consider the following graph.



Say, using Kempe's heuristic, if there is a choice to spill, the node chosen for spilling is derived using one of the two schemes: the one with the (i) highest node number, (ii) smallest node number. Assuming the max number of available colors = 3, the number of nodes that cannot be colored using Kempe's heuristic is

(a) 2 using scheme (i), and 1 using scheme (ii)
(b) 3 using scheme (i), and 1 using scheme (ii)
(c) 2 using scheme (i), and 2 using scheme (ii)
(d) 3 using scheme (i), and 3 using scheme (ii)

17. Consider the following C code:

```
void bar(){
  La: i=1;
  L0: if (i > n) goto L1;
  Lb: j=1;
  Lc: goto L2;
  L1: j=2;
  L2: k = 3;
  Ld: if (n > 0) goto L3
  L4: p++
  Lf: if (p > 10) return;
  Lg: k++;
  Li: goto L4
  L3: if (m > 0) return;
  Lh: k--
  Lk: goto L0
}
```

The number of basic-blocks in the CFG of the above code are:

(a) 6
(b) 8
(c) 10
(d) 11

18. (correct answer: 1 mark; incorrect answer = -0.5) Which of the following is true with respect to basic-blocks?

(a) A basic block can have more than one successor
(b) A basic block can have more than one predecessor
(c) Neither (a) nor (b)
(d) Both (a) and (b)

19. (correct answer: 1 mark; incorrect answer = -0.5) Which of the following is true with respect to three-address-codes?

(a) Each IR instruction must have exactly three operands
(b) Each IR instruction may have at-most three operands
(c) Each IR instruction must have at-least three operands
(d) There can be at most three types of instructions

20. (correct answer: 1 mark; incorrect answer = -0.5) Consider a code transformation shown below:

```
if (a && true)                 if (a)
b = a * 1;           ⇒         b = a;
else                           else
z = c + 0;                     z = c;
```

Name the optimization invoked.

(a) Algebraic simplification
(b) Copy propagation
(c) Strength reduction
(d) Variable elimination

21. (correct answer: 1 mark; incorrect answer = -0.5) Which of the following is true with respect to register allocation?

   (a) If two variables' live-ranges intersect then they may not be assigned the same register
   (b) In the Linear-scan register allocation algorithm, a variable may be assigned different registers at different program points.
   (c) If a variable is "live" at a program point then it has to be defined at that program point.
   (d) None of the above

22. (correct answer: 1 mark; incorrect answer = -0.5) Which of the following is true with respect to runtime-management?

   (a) Local variables can be saved on heap, but not done so for performance reasons.
   (b) Return address must always be passed in a register.
   (c) It would be incorrect for a callee to use all the registers as callee save registers
   (d) All of the above.

23. (correct answer: 1 mark; incorrect answer = -0.5) Consider a code transformation shown below:

```
foo(int p){                              foo(int p){
        int x=3, y=4, z;                         int x=3, y=4, z;
        if (bar(p)) {                            if (bar(p)) {
                z = x + 1;                               z = 4;
        } else {                  ⇒              } else {
                z = y;                                   z = 4;
        }                                        }
        return z;                                return 4;
}                                        }
```

Name the optimization(s) invoked.

   (a) Constant propagation
   (b) Copy propagation
   (c) Strength reduction
   (d) Variable elimination

24. (correct answer: 1 mark; incorrect answer = -0.5) Which of the following is true with respect to parsing?

   (a) LR(1) parsers are more powerful than LL(1) parsers.
   (b) LR(k+1) parser is more powerful than LR(k)
   (c) Left recursion is not problematic for LR parsing
   (d) All of the above

25. (correct answer: 1 mark; incorrect answer = -0.5) Which of the following is true with respect to runtime-management?

   (a) A caller can always know what all registers may be used by a callee at a particular call-site.
   (b) A callee can always know what all registers may be used by its caller, for a call at a particular call-site.
   (c) Neither (a) nor (b)
   (d) Both (a) and (b)