CS6235 Final Exam: May 05 2022

Maximum marks = 45, Time: 3.00 hr, Open Book, Closed Neighbor

Name: _____ Roll: _____

- **Write your roll number on each of the answer sheet.**
- The marks of each question is specified next to the question.
- MCQ questions have no partial marks.
- MSQ questions - will be evaluated only if you have not selected any incorrect answer. For the correct answers - you may get partial credit.

Section 1. MCQ/MSQ

1. **Deadlock detection** [5]
   Consider the following Java code:

```
class Root{
  static void main(String[] a){
    Object sh1 = new Object();    // O1
    Object sh2 = new Object();    // O2
    new Thread1(sh1, sh2).start(); // T1
    new Thread2(sh2, sh1).start(); // T2
  } }
class Thread1 {
  Object l1, l2;
  public Thread1(Object x1, Object x2){l1 = x1; l2 = x2;}
  void run(){
    synchronized (l1)   // L11 {
      synchronized (l2) // L12 {
        System.out.println("Critical section1");
      } } } }
class Thread2{
  Object l1, l2;
  public Thread1(Object x1, Object x2){l1 = x1; l2 = x2;}
  void run(){
    synchronized (l1)   // L21{
      synchronized (l2) // L22{
        System.out.println("Critical section2");
      } } } }
```

   Say we use the unsound and incomplete analysis discussed in the class (based on the paper 'Effective Static Race Detection') on. the deadlock query (d = `T1, L11, L12, T2, L21, L22`). Which of the following statements is true?

   (a) d is not denoted as a deadlock as reachableDeadlock d returns false.

   (b) d is not denoted as a deadlock as aliasingDeadlock d returns false.

   (c) d is not denoted as a deadlock as escapingDeadlock d returns false.

   (d) d is denoted as a deadlock.

2. **Memory consistency model-I.** [5]
Consider the trace of two parallel threads `T0` and `T1` and their memory accesses (in order). As usual, the horizontal axis represents the global time line. The actual values read/written are skipped for brevity.

```
T0:   W(x)   W(y)   W(z)

T1:               R(y)   R(x)
```

Which of the accesses must be marked, at the minimum, as synchronization accesses to realize DRF0 consistency?

(a) either of `x` or `y`, with synchronization edges from the corresponding writes to reads.

(b) `x` only, with synchronization edge from W(x) to R(x).

(c) `y` only, with synchronization edge from W(y) to R(y).

(d) Both `x` and `y`, with synchronization edges from the corresponding writes to reads.

3. **Memory consistency model-II** [5]
Consider the trace of two parallel threads `T0` and `T1` and their memory accesses (in order). As usual, the horizontal axis represents the global time line.

```
// Initially, x = y = 0

T0:   W(x)1   W(x)2  W(y)3   W(x)4

T1:                                R(y)v1   R(x)v2
```

Which of the following values for `v1` and `v2` will not be consistent as per TSO?

(a) `v1 = 0` and `v2 = 1`

(b) `v1 = 3` and `v2 = 2`

(c) `v1 = 3` and `v2 = 1`

(d) `v1 = 0` and `v2 = 4`

4. **Memory consistency-mode-III** [5]
Find below an execution trace. Each line described a different process. The horizontal axis represents the global time line; for instance, as per the global clock, the operation `R(x)2` in process 2 executed after `W(x)2` in process 1.

```
1.   W(x)1 W(x)2 sync
2.              R(x)2 W(x)5 sync
3.              W(x)6 sync
4.                                sync R(x)5, R(x)6
```

Indicate the strongest memory consistency model (among Causal, Eventual and Weak ordering) that can realize the trace shown for process 4. Assume that if P1 executes a `sync` operation after P2 has executed a `sync` operation, then all the writes of P2 are guaranteed to be visible (in program order) to P1 after P1 executes the `sync` operation.

(a) Causal

(b) Eventual

(c) Weak Ordering

(d) None of the above

5. **Data Race.** [5]

Consider the shown trace and answer the following questions: Does the program have a race as per CP ordering based datarace detector and what are the valid CP ordered statements?

```
    T1          T2
L1  w(y)
L2  w(x)
L3  acq(l)
L4  rel(l)
L5              acq(l)
L6              r(x)
L7              r(y)
L8              rel(l)
```

(a) No, L1→L2→L3→L4 and L5→L6→L7→L8

(b) No, L1→L2→L3→L4→L5→L6→L7→L8

(c) Yes, L1→L2→L3→L4 and L5→L6→L7→L8

(d) Yes, L1→L2→L3→L4→L5→L6→L7→L8

6. **Serial program analysis-I.** [5]
Consider a restricted variation of TACoJava, where

i the language supports only one class,

ii only one object of that class can be created,

iii the class can have no fields,

iv the class can have many functions, but the functions can take only integer arguments, and

v no local variable can be assigned more than once.

Which of the following statements is/are true in this context:

(a) Kildall's intra-procedural algorithm can identify all the constants in all the functions.

(b) Context-insensitive constant propagation algorithm can identify all the constants in all the functions.

(c) Context-sensitive constant propagation algorithm can identify all the constants in all the functions.

(d) None of the above.

7. **Serial program analysis-II.** [5]
Consider the following Java code.

```
class A{
 A f;
 void foo(A a){
    f = new A();
 }
 void bar(A x) { x.f = this; } }
class B extends A{
 void foo(A a){
   f = a;
 }
 void bar(A x) { x.f = this;  } }
class Main {
  pubic static void main(){
          A a = new A();
          B b = new B();
          while (*) {
            a.foo(b);
            a.f.bar(a);
        }
  } }
```

Which of the following is/are true with respect to this code?

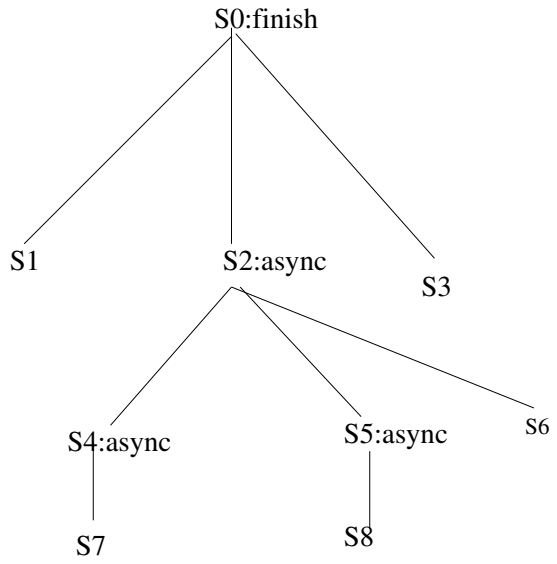
(a) Performing points-to analysis first (assuming CHA as the call graph analysis) and then performing call-graph analysis will lead to imprecise call-graph.

(b) Performing call-graph analysis first (assuming most conservative points-to analysis) and then performing points-to analysis using that call-graph, will lead to imprecise points-to information.

(c) Performing points-to analysis and call-graph analysis alternatively till fixed point will give the most precise points-to and call-graph information.

(d) Performing points-to analysis and call-graph analysis alternatively till fixed point will STILL NOT give the most precise points-to and call-graph information.

8. **MHP Analysis** [5].
Consider a PST as shown below on which we want to perform the incremental MHP analysis studied in the class.



Say, the finish/async nodes are added back in the following order: S0, S2, S5, and S4.
Which of the following is/are true with respect to the MHP maps?

(a) On adding S5 back, only the MHP map of S8 will change.

(b) On adding S4 back, only the MHP maps of S5, S6, S7 and S8 will change.

(c) On adding S2 back, the MHP maps of all the nodes except S0 and S1 will change.

(d) On adding S0 back, only the MHP map of S0 will change.

9. **Intermediate Representation** [5].
Consider the following C code:

```
i = n;
while (1){
   switch (i % 3) {
      case 0: i=i/4; break;
      case 1: i=i/7; break;
      case 2: goto outer;
   }
}
outer:        print i;
```

Draw the CFG in your note book. Don't keep any basic block that has no instructions. The minimum number of basic blocks (excluding the start and end basic blocks) and the back edges in the CFG, respectively, are:

(a) 5, 1

(b) 6, 2

(c) 6, 1

(d) 5, 2