Quiz 2, CS6013

Maximum marks = 40, Time: 50 min

11-Oct-2021

Read all the instructions and questions carefully. You can make any reasonable assumptions that you think are necessary; but state them clearly. There are total seven questions (total 45 marks). Maximum marks that you can get = 40. You will need approximately 6 minutes for answering a 5 marks question (plan your time accordingly). For questions with sub-parts, the division for the sub-parts are given in square brackets.

Start each question on a new page (and write your roll number on each page – both sides of the sheet, that is). Think about the question before you start writing and write briefly. Each question also specifies the maximum number of allowed pages (A4 size) for the question. If the answer for any question is spanning more than specified number of pages, we will strictly ignore the spill-over text. If you scratch/cross some part of the answer, you can use space from the next page.

1. **Control flow Optimizations.** Write a C code, where the control flow optimization "Straightening" can be applied. Show the optimized code. [5]
2. **Optimizations.** Consider the C code shown in Fig. 1. State the optimizations that you may apply on this code and the final optimized code that you will obtain. [5]
3. **Points-to Analysis.** Consider the Java code shown in Fig. 2. Write the points-to maps assuming a flow-insensitive points-to analysis. [5]
4. **Structural Analysis.** Consider a hypothetical language where the *switch* statement has similar semantics as Java, but with one change:
   - There can be no break statements inside switch.

   Write the transfer function for the switch statement [5].
5. **Control Flow Analysis.** For the GCD program shown in Fig. 3 draw the CFG [4], draw the dominator tree [5], and identify the back edge [1].
   Note: A dominator tree (N, E) is a tree where, N=set of all the basic blocks, and $(n_1, n_2) \in E$, if n_1 immediately dominates n_2.
6. **Dependence Analysis.** Draw the dependence graph for the code shown in Fig. 3.
7. **Potpourri.** State true or false. [5]
   (a) Loop inversion can be applied to a for-loop as well.
   (b) You can obtain a non-natural loop using Java code (without applying any optimization).
   (c) Every node has at least one dominator.
   (d) Since context sensitive analysis is more precise than context insensitive analysis, the latter has no use.
   (e) A call graph can have cycles due to the loops in the program.

```
y = 0
if (x > 3) {
  if (x ==  2)
    y = 1
  else
    y = 2;
  print  y
}
print y
```

Figure 1: Sample C code

```
class A{
1  public void m(){
2   X v1 = new X();
3   X v2 = new X();
4   X v3 = null;
5   v1 = new X();
6   v1.f = v2;
7   if (*){
8     v3 =  new X();
9   }
10   v2.f = v3;
  }
}
```

Figure 2: Sample Java code

```
int GCD(){
1     a = input();
2     b = input();
3     if (a == 0)
4         return b;
5     else if (b == 0)
6         return a;
7     while (a != b)
8         if (a > b)
9             a = a - b;
10        else
11              b = b - a;
12    return a;
}
```

Figure 3: GCD code example