Quiz 2, CS6013

Maximum marks = 35, Time: 60 min

31-Oct-2022

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 five questions (total 35 marks). You will need approximately 8 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 "unreachable code elimination" 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. **Constant Propagation.** Write the transfer functions for the statements shown in the Figure 2. [10]
4. **Control Flow Analysis.** For the Bubble sort program shown in Fig. 3 draw the CFG [4], draw the dominator tree [5], and identify the back edges [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.
5. **Potpourri.** State true or false. [5]
   (a) Reaching definitions can be used for identifying dead code
   (b) Every node has at least one post-dominator.
   (c) In a given C code, it may have multiple nodes with no predecessors.
   (d) For every program conditional-constant-propagation finds more constants than simple-constant-propagation.
   (e) The minimum number basic-blocks in any C function (excluding Entry and Exit) is 2.

```
int foo(){
  int y = 0;
  int x = 10;
  if (x < 3 && y > 0) {
    if (x == 10)
      y = 1
    else
      y = 2;
    print  y
  }
  print y
}
```

Figure 1: Sample C code

```
class A{
1.  public void m(int n){
2.    int x = 3;
3.    int y = 5;
4.    x = y + 4;
5.    z = (x > n);
6.    if z goto L1
7.    y = y + 1;
8.L1: return y;
  }
}
```

Figure 2: Sample Java code

```
int bsort(int *A, int n){

    int i, j, sCnt = 0;
    for (i=0; i<n-1 ; ++i)
       for (j=i+1; j<n; ++j)
         if (A[i] < A[j]) {
           int temp = A[i];
           A[i] = A[j];
           A[j] = temp;
           sCnt ++;
         }
    return sCnt;
}
```

Figure 3: Sort code example