CSL862 Minor2 Exam
Advanced Topics in Software Systems
Sem I, 2013-14
October 6, 2013

Answer all 3 questions Max. Marks: 46

1. Consider the following programs A and B:

```
void programA(a, n):
int i;
i = 0;
while (i <  n) {
    a[i] = 2 * i;
    i = i + 1;
}

return;
```

```
void programB(a, n):
int j, k;
j = 0;
k = 0;
while (j < n) {
  a[j] = k;
  j = j + 1;
  k = k + 2;
}
return;
```

a. Write a simulation relation table that can prove that these programs are equivalent. [10]

4 - 8 : k = 2i, j = i
3 - 7 : true
1 - 5 : true

3 marks if CFG is drawn
8 marks if partially correct/has given first constraint
10 marks if everything is correct

b. Show the working of the Scan and Solve procedures to infer the simulation relation table for this program. [18]

Look at the discussed working in class.

6 marks for transfer functions
4 marks for correct recursive scan calls
4 marks for correct MarkRelated calls
4 marks if everything else is correct.

2. Consider the following C program, its corresponding assembly, and pre- and post-conditions:

Program:

```
int main(int a[]) {
  int i;
  for (i = (length(a) - 1)/2; i >= 0; i--) {
    a[2*i] = i;
  }
  return;
}
```

Assembly:

```
#a0 - base address,   a1 - array length
main:
      mov  a1, t0                                  # i = length(a)
 L1: ANN_INV(_____________________________)
      bl    t0, L2                                 # i < 0
      s8addl   t0, a0, t1                          # t1 = a0+8*i
      stl t0, 0(t1)                                # *t1 = i
      subl t0, 1, t0                               # i--
      br  L1                                       # unconditional branch
L2:  ret
```

main : (Pre = a0 : array(int, a1) ^ a1 >= 0,   Post = true)

a. What should be the loop invariant annotation (inside ANN_INV() expression)? Just provide the annotation -- you do not need to discuss how a compiler can automatically come up with such annotations. [4]

t0: int && t0 <= (a1-1)/2, {t0, t1}

b. Write the safety predicate for this program which checks type and memory safety. Assume saferd() and safewr() predicates to have the same semantics as given in the “The Design and Implementation of a Certifying Compiler” paper. [8]

forall a0 forall a1 forall r_m
(a0: array(int, a1) && a1 >= 0) =>
(k: int && k >= k) &&
(forall t0 forall t1
  (t0: int && t0 <= k)
  t0 >= 0 =>
    safewr(a0 + 8*t0, t0) &&
    t0 - 1 : int && t0 - 1 <= k)

where k = (a1 - 1)/2

3. Certified Compilation

a. Briefly explain the construction of a certified compiler from a certifying compiler. [3]

If verification of generated code by certifying compiler is true then output the generated code else Null.

b. Which of the compiler passes have been proved using the “certifying compiler” approach, and why? [3]

Graph coloring: Because of the complexity of the algorithm it is easy to verify code generated instead of verifying the code implementing the pass.