IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2015

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
BEng Honours Degree in Mathematics and Computer Science Part I
MEng Honours Degree in Mathematics and Computer Science Part I
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C141=MC141

REASONING ABOUT PROGRAMS

Tuesday 5 May 2015, 10:00
Duration: 80 minutes

Answer ALL TWO questions

Paper contains 2 questions
Calculators not required

1 This question is about structural induction.

a Consider the datatype `Tree a` defined as follows:

```
data Tree a = Leaf a  | Node (Tree a) (Tree a)
```

Write the inductive principe which implies $\forall$ `t:Tree a`. P(`t`), for some property $P \subseteq$ `Tree a`.

b Consider the datatype `Term` defined as follows:

```
data Term = Val Int | UMinus Term | Mult Term Term
```

Write the inductive principe which implies $\forall$ `t:Term`. P(`t`), for some property $P \subseteq$ `Term`.

c Now consider the following functions:

```
eval :: Term -> Int
eval (Val i) = i
eval (UMinus t) = - (eval t)
eval (Mult t1 t2) = (eval t1) * (eval t2)

rip :: Term -> Term
rip (Val i) = (Val i)
rip (UMinus t) = rip t
rip (Mult t1 t2) =  Mult (rip t1) (rip t2)

pos :: Term -> Bool
pos (Val i) = True
pos (UMinus t) = not (pos t)
pos (Mult t1 t2) = ( (pos t1)==(pos t2) )

wSign :: (Int, Bool) -> Int
wSign (i, True) = i
wSign (i, False) = -i
```

i) Take `trm` to stand for

```
UMinus (Mult (UMinus (Val -3)) (Val 2)).
```

Write out the the result of evaluating the following expressions (but do *not* write any of the intermediate steps):

a. `eval trm`
b. `rip trm`
c. `pos trm`
d. `eval (rip trm)`
e. `wSign (eval (rip trm)) (pos trm)`

ii) Prove the following property

```
∀t:Term. eval t = wSign (eval (rip t)) (pos t).
```

In the proof, state what is given, what is to be shown, what is assumed, which variables are taken arbitrarily, and justify each step.

Prove the base case as normal. In the inductive step, consider *only* the case where `t` has the form `Mult t1 t2`.

You may use the facts that

```
(A):  ∀i,j:Int.  i*(-j) = -(i*j)

(B):  ∀i:Int.∀b:Bool.
          wSign i (not b) = -(wSign i b)

(C):  ∀i1,i2:Int.∀b1,b2:Bool.
          wSign (i1*i2) (b1==b2) =
          (wSign i1 b1)*(wSign i2 b2)
```

The three parts carry, respectively, 10%, 15%, and 75% of the marks.

2 This question is about loop invariants:

Consider the following Java method `smooth` which performs a simplified smoothing (local averaging) of the contents of an array `a`:

```
 1 void smooth(int[] a){
 2 // PRE: $a \neq null \land ???$
 3 // POST: $\forall k \in [1..a.length - 1).\ a[k] = \frac{a_0[k-1]+a_0[k]+a_0[k+1]}{3}$
 4     int i = 0;
 5     int prv = 0;
 6     // INV: ??? ∧ ??? ∧ ??? ∧ ???
 7     while (i < (a.length - 1)) {
 8         int cur = a[i];
 9         int nxt = a[i+1];
10         a[i] = (prv + cur + nxt) / 3;
11         prv = cur;
12         i++;
13     }
14     // MID: $i = a.length - 1 \land \forall k \in [1..i).\ a[k] = \frac{a_0[k-1]+a_0[k]+a_0[k+1]}{3}$
15     a[i] = (prv + a[i]) / 3;
16 }
```

a Write the final state of the array $a =$ `[5,9,2,4]` after running `smooth(a)`.

b The precondition `PRE` of the `smooth` method is incomplete.

i) Give an example integer array $b \neq$ `null` where running `smooth(b)` would throw an exception.

ii) Identify the line on which this exception would be thrown.

iii) Briefly describe why this exception would be generated.

iv) Complete the precondition `PRE` so that it rules out your example from part b(i) above.

c Complete the loop invariant `INV` so that it is appropriate to show total correctness. (You do not need to prove anything.)

[**Hint:** *There are four conjuncts. The first should bound* `i`*, the second should define* `prv`*, and the third and fourth should describe the state of the array* `a`*.*]

d Prove that all of the array accesses within the `smooth` method are in bounds. State clearly what is given and what you need to show in each case.

(You may assume that the length of the array `a` is not changed by the loop.)

e The `smooth` method performs an in-place update on the array `a` that it is called on. Imagine that the `smooth` method were run over the same array a large number of times (say several billion). What would happen to the contents of this array?

[**Hint:** *Look carefully at the code and notice that the method's specification does not describe what happens to the first or last elements of the input array.*]

The five parts carry, respectively, 10%, 20%, 20%, 40%, and 10% of the marks.