UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1996

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
BSc Honours Degree in Mathematics and Computer Science Part I
MSci 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 Royal College of Science
Associateship of the City and Guilds of London Institute

PAPER 1.2 / MC1.2

REASONING ABOUT PROGRAMS

Tuesday, April 30th 1996, 4.00 - 5.30

Answer THREE questions

For admin. only: paper contains
4 questions
3 pages (excluding cover page)

1 This question asks you to develop a function called Eqno, with the following informal specification.

Given arrays A, B of integers, with lower(A) = lower(B) = 0 and upper(A) = upper(B), Eqno should return the number of indices i such that A(i) = B(i).

a Give a suitable pre-condition and post-condition for Eqno.

b The function Eqno could be written using a loop. Draw a diagram of A, B, showing suitable pointers, etc., representing the situation at the beginning of an arbitrary cycle in the execution of the loop.

c Write the body of the Eqno function. Include the loop variant and invariant as comments.

d Prove that the loop code re-establishes the loop invariant, and that, if the loop terminates, the post-condition is set up. Remember to check that all array accesses are legal.

The four parts carry, respectively, 20%, 20%, 30%, 30% of the marks.

2a What is a *tail recursive function?* What is an *accumulating parameter,* and how is it related to tail recursion?

b A recursive procedure that calculates the square of a non-negative integer is given below.

```
function Sq(n : int) : int
% pre  n >= 0
% post r = n*n
      if n=0 then result 0
      else result Sq(n-1) + 2*n - 1
      end if
end Sq
```

i) Give the definition of a tail recursive function TRsq that, with a suitable choice of value for the accumulating parameter, produces the same result as Sq. Your answer may be in either Turing or Miranda. It should work by adding 2*n – 1 into the accumulating parameter.

ii) What arguments must you give the TRsq function to calculate Sq n?

c Write the Turing code of a procedure LPsq that calculates TRsq by using a loop, without recursion. Do not forget to include a pre-condition, post-condition, loop variant, and loop invariant as comments.

d Prove by induction that the function TRsq that you defined in part b(i), when given the parameters that you specified in part b(ii), does produce the same result as Sq n.
Warning: you will have to formulate an inductive hypothesis that deals with arbitrary values of the accumulating parameter.

The four parts carry, respectively, 20%, 15%, 35%, 30% of the marks.

3 The binary chop algorithm, implemented in Turing, is as follows.

```
function chop (A: array of int, x: int):int
% Pre:
% Post: 0 ≤ r ≤ upper(A)+1
%            & (A)i:int.(0≤i<r ⇒ A(i) < x & ...

var Left: int := 0
var Right : int := upper(A)+1
var Middle : int

    loop
        % Invariant:
        % Variant:
        exit when Left >= Right
            Middle := (Left+Right) div 2
            if A(Middle) < x
        then Left := Middle+1
            else Right := Middle
            end if
    end loop
    result Left
end chop
```

a Complete the pre-condition, post-condition, loop variant, and loop invariant for chop.

b Prove that each iteration of the loop re-establishes the invariant and decreases the variant.

c Using only that the function chop meets its post-condition, prove informally that $\forall x,y.\text{int}.(x \leq y \rightarrow \text{chop}(A,x) \leq \text{chop}(A,y))$. (You might want to assume the property fails, and get a contradiction.)

Turn over ...

4a Consider the following Miranda definitions, used to convert natural numbers into lists of binary digits:

```
bitlist n
      = [],        if n = 0
      = bitlist ((n div 2)++[n mod 2]),  otherwise

bitacc n bs
      = bs,        if n = 0
      = bitacc (n div 2) ((n mod 2):bs),  otherwise
```

Prove (using 'course of values' induction on n) that if n is a natural number then

```
bitacc n bs  =  (bitlist n)++bs.
```

You may assume any properties of ++ that you need, such as associativity, as long as you state them clearly.

b A binary tree can be represented by the Miranda data type

```
btree * ::= Emptytree | Node (btree *) * (btree *)
```

Consider the following functions on btrees:

```
leaves::(btree *)->[*]
leaves Emptytree = []
leaves (Node t1 x t2)
      = [x],  if t1 = Emptytree & t2 = Emptytree
      = (leaves t1) ++ (leaves t2),          otherwise

revtree::(btree *)->(btree *)
revtree Emptytree = Emptytree
revtree (Node left y right)
      = (Node (revtree right) y (revtree left))
```

Suppose that the function reverse::[*]->[*] produces a list with elements in reverse order.

Show, by induction on the structure of t, that for any btree t

```
leaves (revtree t)  =  reverse (leaves t)
```

You need the following properties of reverse, which you may use without proof:

```
reverse [] = []
reverse (xs++ys) = (reverse ys)++(reverse xs)
```

Hint: In the inductive step , treat separately the special case where both subtrees are empty.

The two parts carry, respectively, 40% and 60% of the marks.

End of paper