UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1997

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 22nd 1997, 4.00 - 5.30

Answer THREE questions

For admin. only: paper contains 4 questions

1a State how you would prove a property P to be true of all natural numbers (0,1,2,...) by simple induction.

b State how you would prove a property Q to be true of all lists of a given type [*], by list induction.

c Prove by simple induction that for any real number $x \neq 1$ and any natural number n,

$$1 + x + x^2 + \ldots + x^n = \frac{x^{n+1} - 1}{x - 1}$$

d Define Miranda functions as follows:

```
dup : * -> [*] -> [*]
    dup x []     = []
    dup x y:ys   = y : (y : dup x ys),      if x = y
                 = y : dup x ys,          otherwise

count : * -> [*] -> nat
    count x []     = 0
    count x y:ys   = 1 + count x ys,   if x=y
                   = count x ys,       otherwise
```

Prove by list induction that for all `x, ys,`

```
count x (dup x ys)    =    2 * count x ys,
```

where '*' denotes natural number multiplication.

It is important to lay out your proofs in parts c and d clearly.

The four parts carry, respectively, 15%, 15%, 30%, 40% of the marks.

2 This question asks you to develop a function called `Jump`, with the following informal specification.

Given a sorted (ordered) array `A` of integers, `Jump` should return the number of different entries in `A`. This is one more than the number of indices `i` such that `lower(A) < i ≤ Upper(A)` and `A(i) > A(i-1)`.

a Give a formal pre-condition and post-condition for `Jump`.
You may use the notation '#i:P(i)' for 'the number of i having the property P(i)'. (Eg. #i:int:(4 ≤ i ≤ 6)=3.)

b The function `Jump` could be written using a loop. Draw a diagram of `A`, 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 `Jump` 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, 15%, 20%, 35%, 30% of the marks.

3 The specification of the binary chop algorithm, in Turing, is as follows.

```
function chop (A: array of int, x: int):int
% Pre:  A is ordered (sorted)
% Post: lower(A) ≤ r ≤ upper(A)+1
%        & ∀i:int.(lower(A) ≤ i < r ⇒ A(i) < x
%                 & r ≤ i ≤ upper(A) ⇒ A(i) ≥ x)
```

a Write the code for `chop`. (The main loop should reset the values of pointers `Left`, `Right`.) Do not forget the loop variant and loop invariant.

b Verify that each iteration of your loop code re-establishes the invariant. You may assume properties of `div` such as `a < b ⇒ a ≤ (a+b) div 2 < b`.

c Which of the following statements are true for all sorted arrays `A` of integers and all integers `x, y`?

i) `chop A x` returns the index of the first occurrence of `x` in `A`.

ii) `chop A (x+y) = chop A x + chop A y`.

iii) If all entries in `A` are equal then `chop A x` is either `lower A` or `upper(A)+1`.

iv) If `x` does not occur in `A` then `chop A x` gives an error.

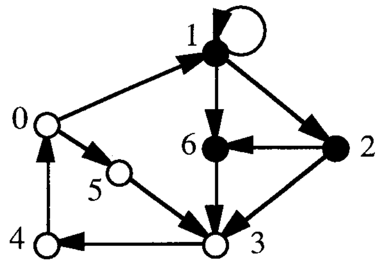
v) If `x` does not occur in `A` then `chop A x = chop A (x+1)`.

d Using only that the function `chop` meets its post-condition, prove informally that for any sorted array `A` and integer `x`, `chop(A,x) ≤ chop(A,x+1)`.

(You might want to assume the property fails, and get a contradiction.)

Turn over ...

4 A *piebald graph* is a directed graph whose nodes are coloured black or white. Eg:



A *shaded path* between two nodes x, y is a path from x to y, all of whose transit (intermediate) nodes, if any, are black. Eg, above, the path 1,2,3 is a shaded path from 1 to 3, because 2 is black. The path 0,1 is shaded because it has no transit nodes. The path 2,3,4 is not shaded, because 3 is white.

a i) Is there a shaded path from 0 to 3 above?

ii) Name two nodes above with more than one shaded path between them.

iii) Name two nodes with an ordinary path between them, but no shaded one.

It is desired to adapt Warshall's algorithm to determine which nodes of an arbitrary piebald graph have a shaded path between them. The graph has nodes `0,1,...,N` and is represented by the two arrays `Edges` and `Black`, where:

`Edge(x,y)` = **true** when there is an edge from `x` to `y`, and
`Black(x)` = **true** when `x` is a black node.

Here is part of the code (omissions are marked by `***`):

```
type Colour : array 0..N of bool
type Adj : array 0..N,0..N of bool

function ShadedPath(Edge: Adj, Black: Colour) : Adj
% Post: ∀x,y:int[0≤x≤N & 0≤y≤N →
%        (r(x,y) ↔ there is a shaded path from x to y)]
var i: int := 0
var Path : Adj := ***
    loop  % variant ***, invariant: 0≤i≤N+1 & ***
    exit when  ***
        ***
        Path(x,y) := Path(x,y) or
                     (Path(x,i) and Black(i) and Path(i,y))
        ***
    end loop
    result Path
end ShadedPath
```

b Complete the code and give a suitable loop invariant and loop variant.

c Prove that the loop code re-establishes your invariant.

d Assuming that the loop terminates, prove that the post-condition is set up.

The four parts carry, respectively, 15%, 30%, 30%, 25% of the marks.

End of paper