UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

# EXAMINATIONS 2001

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

## PAPER C121=MC121

## PROGRAMMING

Friday 18 May 2001, 14:30
Duration: 120 minutes

*Answer FOUR questions*

Paper contains 6 questions
Calculators not required

***Section A*** *(Use a separate answer book for this Section)*

1 This question is to be answered in the programming language Turing.

a If one looks at the square numbers and the differences between each adjacent pair of square numbers a pattern emerges.

squares: 0 1 4 9 16 25 36 49...

differences: 1 3 5 7 9 11 13...

Write a program to print out the first 100 square numbers, ten to a line, *without* using multiplication `*` or exponentation `**`.

b i) Write a function called `absDiff`, which takes two integers as parameters and returns the absolute value of the difference between the two integers. Your function *must not* call any inbuilt Turing function such as `abs`.

ii) Write a function called `intRoot`, which takes an integer as a parameter and returns the integer value closest to the square root of the parameter. Your function *must not* call any inbuilt function, but may call `absDiff`.

iii) Given a sequence where the $i^{th}$ term is the closest integer to $\sqrt{i}$ (square root of i), write a procedure called `putSequence`, which takes an integer `n` as a parameter and prints on the screen the first n integers in the sequence, ten to a line. You may use previously written and inbuilt functions and procedures. So `putSequence(10)` would print on the screen:

```
1  1  2  2  2  2  3  3  3  3
```

iv) Write a function `numFound` that takes two integers `n` and `m` and returns the number of occurrences of `m` in the sequence of the first `n` integer square roots. So

```
numFound(10,3) is 4
numFound(10,1) is 2
numFound(10,5) is 0
```

In parts (iii) and (iv) you may use previously written and inbuilt functions and procedures.

*The two parts carry, respectively, 25% and 75% of the marks.*

2 Many board games are played on an 8 by 8 chessboard. This question is about writing functions for processing coordinates representing board positions in programs for such games. (Note that you are not asked to get input from the keyboard.)

This question is to be answered in the programming language Turing. Where appropriate include *pre-conditions* and *post-conditions*. These may be written in unambiguous English, logic or Turing.

a i) Write a predicate function `isNum` whose input parameter `c` is a character. The predicate should return `true` if and only if `'1'≤c≤'8'`.

ii) Write a predicate function `isChar` whose input parameter `c` is a character. The predicate should return `true` if and only if `'a'≤ c≤'h'`.

iii) Write a function `convert`, which converts a character in the range `'1'≤c≤'8'` to the number it represents. Thus `convert('3')` returns `3`.

b Write a predicate function `isValid` which takes two characters as parameters and returns true if and only if the first is in the range `'1'` to `'8'` and the second in the range `'a'` to `'h'` or if the first is in the range `'a'` to `'h'` and the second in the range `'1'` to `'8'`.

c i) Declare a type `coordinates` which holds a number and a character.

ii) Write a function (or procedure) `package` that takes as input parameters two characters, which are known to be valid (as defined in part b of this question), and returns a variable of type `coordinates`, which holds the input parameters.

*The three parts carry, respectively, 40%, 20% and 40% of the marks.*

**Section B** (Use a separate answer book for this Section)

3a Define the *Abstract Data Type* (ADT) *Binary Search Tree* (BSTree), and list its *Access Procedure Headers* with their pre- and post-conditions.

Explain the advantages and disadvantages of using a Binary Search Tree compared with an ordered linked list data structure; you should consider flexibility and efficiency in your comparison.

b Describe briefly two modifications of the ADT Binary Search Tree whose purpose is to improve on its performance, and in each case explain how an improvement is achieved.

c Write the Turing code for the following traversal functions:

```
function InOrder(T:BSTree):List
%pre: takes a BST, T, of integer items
%post: returns an ordered list of the items in T, by inorder traversal
```

```
function LevelOrder(T:BSTree): List
%pre: takes a BST, T, of integer items
%post: makes a level order traversal of T: visits first the root, then the child
%nodes of the root, then the child nodes of the root's children, in the order in
%which their parent nodes were visited, and so on. Returns a list of the items
%in the order in which they were visited.
```

In writing your answer to part c you may assume the availability of ADTs *List, Queue* and *Stack*, with integer data type, and their standard access procedures. Any necessary additional procedures should be given in full.

*The three parts carry, respectively, 30%, 30%, 40% of the marks.*

4 A programmer must choose an implementation of the function:

```
function Fibonacci(n:int): int
%pre: takes a non-negative integer
%post: returns f_n where f_0=0, f_1=1, f_i=f_{i-1}+f_{i-2}, i≥2
```

You are asked to produce a document which will inform this choice, and which contains the following components:

a written in Turing:

i) a *recursive* implementation of Fibonacci

ii) an *iterative* implementation of Fibonacci, without using any other data structure

iii) an *iterative implementation which mimics the action of the recursive method* by use of an Abstract Data Type (ADT) *Stack.* You may assume the availability of ADT Stack with the standard access procedures.

b written in English:

A paragraph in which you compare your versions of the function Fibonacci in your answer to part a, including consideration of space and time resources needed, and ease of testing and maintenance.

c given the definition of *Generalised Fibonacci numbers, $f^m_i$ of order m*:

$$f^m_0 = f^m_1 = \ldots = f^m_{m-2} = 0, \quad f^m_{m-1} = 1, \quad f^m_i = \sum_{j=i-m}^{i-1} (f^m_j), \quad i \geq m, \; m \geq 2,$$

and using one of the methods in part a, write the Turing code for the function:

```
function GenFibonacci(m,n:int): int
%pre: takes m≥2, n≥0
%post: returns f^m_n, the nth Fibonacci number of order m.
```

*The three parts carry, respectively, 40%, 25%, 35% of the marks.*

5 This question concerns the *dynamic implementation* of an Abstract Data Type (ADT) ErdosStruct.

The mathematician Paul Erdos holds the record for the largest number of fellow researchers with whom he published results.

The Erdos numbers are defined as follows: people with whom Erdos jointly published a paper are said to have the Erdos number 1; people who have published jointly with someone whose Erdos number is 1 have Erdos number 2, and so on. People who do not have an Erdos number which is $\leq 7$ are said to have Erdos number $\infty$.

The ADT ErdosStruct has been defined to facilitate the storage of data about people who have Erdos number $\leq 7$. The access procedures are:

**function** EmptyE(): ErdosStruct
%post: returns an empty ErdosStruct

**function** IsEmptyE(E:ErdosStruct): **boolean**
%post: returns true if E is empty, false otherwise

**function** FindENo(P:Person, E:ErdosStruct)**: int**
%pre: takes a Person and an ErdosStruct
%post: returns the smallest Erdos number of P; returns 0 if no p, $1 \leq p \leq 7$

**procedure** AddPublication(P,Q:Person, **var** E:ErdosStruct)
%pre: takes P and Q who have published together, and E
%post: adds joint publication of P and Q to E.

For your chosen implementation:

a draw a diagrammatic representation of the ADT ErdosStruct containing the following data, where each bracketed pair of names represents joint publication:

(Jones, Erdos), (Smith, Jones), (Wong, Jones), (Lim, Erdos),
(Lee, Wong), (Jake, Erdos), (Brown, Jake), (Black, Lee).

b write in Turing all the necessary type definitions

c write Turing code for the access procedures:

i) `EmptyE`

ii) `AddPublication`

*The three parts carry, respectively, 25%, 25%, 50% of the marks.*

6a Describe the concepts: *object*, *class,* and *inheritance*. Explain the difference between *object-oriented* and *procedure-oriented* programming.

b A **pict** class, implemented by the OOT program pict.tu, has the following methods:

Position(x,y:int) : sets current position to (x,y)

Drawtree(x,y,b,h:int): draws a tree shape as in fig. 1, consisting of a green triangle and a brown rectangle.

Drawbubble(x,y, r:int): draws a blue circle centred at (x,y), radius r, with a red square shape inside it, as in fig. 2.

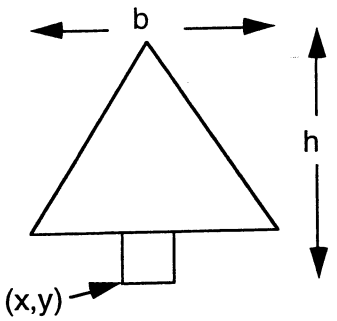


fig. 1

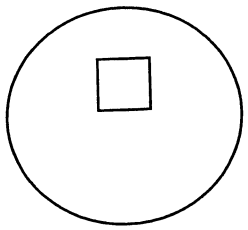

fig. 2

i) write an OOT program *pict.tu* to implement the class pict. You may assume the availability of the OOT predefined drawing routines, which include:

Draw.Line((x1,y1,x2,y2,c):draws line in colour c from (x1,y1) to (x2,y2)

Draw.Box(x1,y1,x2,y2,c) : draws rectangle in colour c, with bottom left corner at (x1,y1) and top right corner at (x2,y2).

Draw.Oval(x,y,xr,yr,c) : draws an oval in colour c centred at (x,y) with horizontal and vertical distances xr and yr respectively from the centre to the oval.

ii) write the implementation of a class **treepict** by inheriting the class pict and adding to it a new method:
DecoratedTree(x,y,b,h:int): draws a tree decorated with shapes drawn using Drawbubble. State any assumptions you make.

*The two parts carry, respectively, 25%, 75% of the marks.*