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
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 1.8

MIRANDA AND PROLOG PROGRAMMING

Friday, May 9th 1997, 2.00 - 3.30

Answer THREE questions

For admin. only: paper contains 4 questions

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

1 A Pythagorean triple is three positive integers *a*, *b* and *c* such that $a^2 + b^2 = c^2$. This question is about producing Pythagorean triples in Miranda. You may assume that all functions you have written in earlier parts of this question are available for use throughout the question. Each function must include both a declaration and definition.

a Write a function `isPythag` that takes a triple of numbers and returns True if the triple is a Pythagorean triple, and False otherwise. So

```
isPythag (6,8,10) is True
isPythag (5,3,4) is False
```

b Write a function `makePairs` which takes a positive integer `n` and a list `l` of positive integers (all unique) and and returns a list of of all pairs where one element of the pair is `n` and the other element of the pair is from the list. If `n` is in list `l` then the pair `(n,n)` should only appear in the resultant list once. So

```
makePairs 3 [1,2,3] is
[(1,3),(3,1),(2,3),(3,2),(3,3)]
```

c Write a function `pairs` which takes a number `n` and returns a list of of all pairs `(p,q)` where $1 \le p \le n$ and $1 \le q \le n$. Each pair should appear in the resultant list exactly once. (In Miranda the notation for the list of the first `k` natural numbers is `[1..k]`.) So

```
pairs 2 is [(1,1),(1,2),(2,1),(2,2)]
```

d Write a function `triples` which takes a positive integer `r` and a list `l` of pairs of numbers and returns a list of of all triples `(p,q,r)` where `(p,q)` is in `l` such that `(p,q,r)` is a Pythagorean triple. So

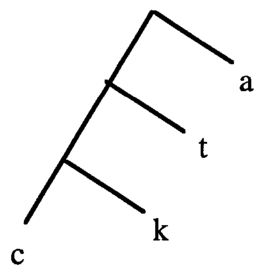
```
triples 5 [(1,2),(3,4),(2,1),(4,3)]  is
[(3,4,5),(4,3,5)]
```

e Write a function `pythagoras` which takes a number `n` and returns all the Pythagorean triples `(p,q,r)` where $1 \le p \le n$, $1 \le q \le n$ and $1 \le r \le n$. So

```
pythagoras 6 is [(3,4,5),(4,3,5)]
```

The five parts carry, respectively, 10%, 25%, 20% 25% and 20% of the marks.

2 The standard ASCII code uses 7 bits to represent each character, so a string with 6 characters requires 42 bits. Huffman coding uses variable length codes; the 6 character string below only requires 12 bits! The code is defined by a binary tree with characters at its leaves: the Huffman Coding Tree. The code for a particular character, c, is a sequence of binary values (Left or Right) describing the path in the tree to the leaf containing c. For example using the tree:



the string "attack" would be coded as:

[Right, Left, Right, Left, Right, Right, Left, Left, Left, Left, Left, Right]

Answer the following questions using the types:

```
huffman ::= Tip char | Bin huffman huffman

step ::=  Left | Right

path == [step]
```

for the Huffman Coding Tree, a binary value indicating one step through the tree, and a path (sequence of steps).

a Define `hcode` which takes a coding tree and a character string and produces the code for the string. You may find it helpful to define auxiliary functions:
```
    codechar :: huffman -> char -> path
```
and
```
    memb :: huffman -> char -> bool
```
which find the code for a single character and check to see if a character appears in a tree, respectively. You may use Miranda's predefined function:
```
    concat :: [[*]] -> [*]
```
which "flattens" a list of lists, in your answer.

b Define `hdecode` which takes a coding tree and the code of a string and produces the decoded character string.

c The Huffman Coding Tree for a particular string is designed so that the most frequently occuring characters have the shortest codes. Using `foldl` write a function `frequency` which takes a character string and a character and returns the number of times the character appears in the string.

The three parts carry, respectively, 50%, 30%, 20% of the marks.

Turn over ...

SECTION B

3 This question concerns the manipulation of symbolic representations of polynomials in a single variable x, such as $bx^2+2x+3c$, in which the only operations are addition, multiplication and raising x to a numeric power.

Any such polynomial can be represented by a Prolog term as follows:

let a(E1, E2) represent the addition of E1 and E2
let m(E1, E2) represent the multiplication of E1 by E2
let p(E1, E2) represent the raising of E1 to the power E2

Thus, $bx^2+2x+3c$ can be represented by a(a(m(b, p(x, 2)), m(2, x)), m(3, c)).

a Write a Prolog program **P1** defining the relation diff(E, D) which holds when

E represents a polynomial, and
D represents the result of differentiating that polynomial with respect to x.

For instance, the query ?diff(p(x, 2), D) might return D as m(2, p(x, 1)), representing the differentiation of x^2 to give 2x. You may freely use Prolog primitives such as \=, atom, is, etc.

Hint — ignore the potential for algebraic simplification, such as replacing p(x, 1) by x. Just express how to differentiate the assumed species of terms. For instance, the differentiation of an addition is expressible by

```
diff(a(E1, E2), a(D1, D2)) :- diff(E1, D1), diff(E2, D2).
```

In fact, your entire **P1** requires only five clauses for "diff".

b Using **P1**, sketch the evaluation of the query ?diff(m(b, p(x, 2)), D), showing carefully how the answer for D is constructed.

c Many terms representing polynomials can be simplified by exploiting arithmetic and basic laws of algebra. For instance, we might simplify a(2, 3) to 5, or simplify a(0, p(x, 1)) to p(x, 1) and then simplify this in turn to x.

Write a Prolog program **P2** defining the relation simp(E, S) which holds when

E represents a polynomial and takes the form a(E1, E2),
  where E1 and E2 represent any polynomials, and
S represents a simplification of that polynomial.

Note that an addition can be simplified when it has an operand 0 or when both its operands are numbers or when both its operands can be simplified. These are the **only** kinds of simplification that **P2** is required to perform.

The three parts carry, respectively, 50%, 20% and 30% of the marks.

4a Write a Prolog program for the relation distincts(L, S, D) which holds when D is a list comprising both the members of list S and one occurrence of each member of L that is not a member of S.
For instance, the query ?distincts([c, a, b, b, a], [], D) might return D=[b, a, c].
Hint — use the second argument to accumulate the distinct members found in the course of deconstructing the first argument.

b Write a Prolog program for the relation occurs(U, L, K, N) which holds when N is the sum of K and the number of occurrences of the member U in the list L.
For instance, the query ?occurs([b, [c, a, b, b, a], 0, N) should return N=2.
Hint —use the third argument to accumulate a count of the occurrences found so far.

c Write a Prolog program for the relation counts(D, L, C) which holds when, for each member U of D, the list C has a member [N, U] where N is the number of occurrences of the member U in L. The program may refer to the "occurs" relation.
For instance, the query ?counts([b, a, c], [c, a, b, b, a], C) should return C=[[b, 2], [a, 2], [c, 1]].

d Write a single clause defining minocc(L, U) which holds when U is a member of L having a minimal number of occurrences in L.
For instance, the query ?minocc([c, a, b, b, a], U) should return U=c.
Hint — use the relations defined earlier to extract the distinct members of L and then to count their occurrences in L. To find the desired member U, exploit the Prolog primitive sort(X, Y) which returns Y as the result of sorting the given list X into ascending lexicographical order.

e By using the Prolog primitives "findall" and "length", rewrite the clause defining "minocc" so that its body comprises **only** Prolog primitives.

The five parts carry, respectively, 25%, 25%, 15%, 15% and 20% of the marks.

End of Paper