PAPER CODE NO. COMP205	EXAMINER : **Dr GR Malcolm** DEPARTMENT : Computer Science Tel. No. **46794**

THE UNIVERSITY
of LIVERPOOL

SEPTEMBER 2002 EXAMINATIONS

Bachelor of Arts : Year 2
Bachelor of Arts : Year 3
Bachelor of Engineering : Year 2
Bachelor of Science : Year 1
Bachelor of Science : Year 2

COMPARATIVE PROGRAMMING LANGUAGES

TIME ALLOWED : Two Hours

INSTRUCTIONS TO CANDIDATES

Attempt ***ALL*** questions in Section A
Attempt ***FIVE*** question from Section B

If you attempt to answer more than the required number of questions (in any section), the marks awarded for the excess questions will be discarded (starting with your lowest mark).

THE UNIVERSITY
of LIVERPOOL

Section A

Each question in this section is worth 4 marks. Answer all questions in this section.

1. For each of the following programming languages, state whether the language is an imperative, functional, or logic programming language:

   (a) Fortran

   (b) Haskell

   (c) Pascal

   (d) Prolog

2. Give three major characteristics of the imperative paradigm.

3. Give three major characteristics of the declarative paradigm.

4. Consider the following fragment of C code:

```
int *p, n=0;
p = &n;
n++;
printf("%d, ", *p);
*p = (*p)++;
printf("%d",n);
```

   What would you expect the output to be?

THE UNIVERSITY
of LIVERPOOL

5. In Ada, formal parameters can be qualified by the keywords `in` and `out`. What is the difference between `in`-parameters and `out`-parameters?

6. What, briefly, is the difference between compiled and interpreted programming languages?

7. In Haskell, `Int` denotes the type of 32-bit integer numbers. Give similarly brief characterisations of the following Haskell types:

   (a) `Char`

   (b) `[Int]`

   (c) `[ (Int, Char) ]`.

8. Consider the following Haskell definitions:

```
dataList = 5 : 2 : []
length [] = 0
length (x:xs) = 1 + length xs
```

   Give a step-by-step reduction of `length dataList`.

9. Consider the following expression, using the list-comprehension notation of Miranda and Haskell:

```
[ (x, y) | x <- [1], y <- [0..x] ]
```

   What would you expect the result of evaluating this expression to be?

10. Consider the following Horn-clause declarations:

```
parent(mark, john).
parent(mark, lucy).
male(mark).
male(john).
brother(X,Y) :- male(X), parent(Z,X), parent(Z,Y).
```

    Give one possible solution to the query `?- brother(A,B)`?

Section B

Each question in this section is worth 12 marks. Answer five questions from this section.

1. (a) What are the potential advantages of enumerated types? **[3 marks]**

   (b) Briefly describe how enumerated types are implemented in C. **[4 marks]**

   (c) In any language you are familiar with (imperative or functional), define an enumerated type to represent the days of the week, and write a function that takes a day of the week as argument and returns a string denoting the *following* day of the week (cyclically, so that Monday follows Sunday). **[5 marks]**

2. (a) Briefly describe the difference between a *formal* and an *actual* parameter. **[3 marks]**

   (b) What is meant by 'call-by-value parameter-passing'? **[3 marks]**

   (c) C implements a call-by-value parameter-passing mechanism. Describe how you can simulate call-by-reference parameter-passing in C, illustrating your answer by writing a C function that takes two integers as parameters, and assigns the larger of the two values to the first parameter. **[6 marks]**

3. (a) In C, define a data type `b_tree` of binary trees with internal labels; i.e., each tree element comprises:

   - an item `value` of type `int`,
   - an item `leftBranch` of type `b_tree`, and
   - an item `rightBranch` of type `b_tree`.

   [6 marks]

   (b) Write C code that will create a binary tree with:

   - `value` equal to `1`,
   - `leftBranch` set to `NULL`, and
   - `rightBranch` a binary tree with `value` set to 2. **[6 marks]**

4. (a) Suggest an appropriate data structure that can be used to store tables of data such as the following:

2	4	6	9	15
16	12	3	1	1
0	0	21	7	3
12	12	15	16	11

[3 marks]

(b) Suppose it was desired to add up all the rows and columns in such a data structure; what sort of data structure could be used to store the results? **[2 marks]**

(c) Give an algorithm that adds up the rows and the columns, and computes the total value of the numbers stored in the table. **[7 marks]**

5. (a) A *rose tree* over a type `a` is a tree with internal labels and an arbitrary branching factor: each rose tree consists of

- an internal label of type `a`, and
- a list of subtrees (each of which is a rose tree).

Define a polymorphic data type of rose trees over a parameter type `a`. **[5 marks]**

(b) Define a higher-order 'map' function that takes a function and a rose tree as arguments, and applies the function to each internal label in the rose tree. Include a type declaration of the function in your answer. **[7 marks]**

6. (a) What is a constructor in Haskell? **[3 marks]**

(b) What is meant by a *pattern* in Haskell? **[3 marks]**

(c) What is *pattern-matching*? Illustrate your answer by describing how the expression `('c',[1,2,3])` matches the pattern `(x, m:n:ns)`. **[6 marks]**

7. (a) Consider the following Haskell definitions:

```
nats = 0 : map (1+) nats

zip [] ys  =  []
zip xs []  =  []
zip (x:xs) (y:ys)  =  (x,y) : zip xs ys
```

Use `zip` and `nats` (note that `nats` is the infinite list `0, 1, 2, ...`) to define a function

```
numberLines : [[Char]] -> [(Int,[Char])]
```

that takes a list of strings, where each string is to be thought of as a line, and returns a list of pairs, where each line is given a number. For example, given the list

```
["i = 0;", "x = 1;", "y = 1;"]
```

`numberlines` should return the list

```
[(0,"i = 0;"), (1,"x = 1;"), (2,"y = 1;")]
```

[4 marks]

(b) Briefly describe the 'topmost-outermost' reduction strategy and say how it implements lazy evaluation. Illustrate your answer by giving a step-by-step evaluation of

```
numberLines ["i = 0;", "x = 1;"] .
```

[8 marks]

8. (a) What is meant by α-conversion in the λ-calculus? **[3 marks]**

(b) What is meant by β-reduction in the λ-calculus? **[4 marks]**

(c) Give a step-by-step reduction of the λ-term

$$\lambda y.((\lambda x.(\lambda y.(x\ y)))\ (\lambda z.(z\ y)))$$

indicating whether each step is an instance of α-conversion or of β-reduction. **[5 marks]**