UNIVERSITY OF SWAZILAND

FINAL EXAMINATION 2006

Title of paper: PROGRAMMING LANGUAGES

Course number: CS343

Time allowed: Three (3) hours

Instructions: Answer any five (5) of the six (6) questions.

This examination paper should not be opened until permission has been granted by the invigilator.

Question 1

a) Describe any two kinds of routine polymorphism.

[4]

b) Distinguish between statements and expressions in source code.

[4]

c) Give an overview of the methods of parameter passing.

[12]

Question 2

a) Give any two reasons for having types in languages.

[4]

b) Explain, with reasons, the facilities that languages should offer in support of programming-in-the-large.

[10]

c) Write a C++ function containing 2 local variables:

- *s*, a static extent variable.
- *d*, a dynamic (or indefinite) extent variable.

Clearly indicate the locations in your function where memory for each variable is allocated and deallocated.

[6]

Question 3

a) Explain, and give reasons for, the main prescriptions ('good practices') of structured programming.

[8]

b) Describe the main characteristics of the object oriented programming paradigm.

[12]

Question 4

a) Define the syntax of expressions in the lambda calculus.

[12]

b) Outline the process by which Prolog's inference engine finds answers to queries containing unbound variable terms.

[8]

Question 5

Define the following functions in Haskell.

a) A function that, given a list of integers, returns their average as a floating point number. In addition, write the type signature of this function.

[4]

b) Ackerman's function, *a(x, y)* with integer parameters, where:
a(x, y) =

• *y+1*	if x=0, y≥0
• *a(x-1, 1)*	if x>0, y=0
• *a(x-1, a(x, y-1))*	otherwise

In addition, write the type signature of this function.

[5]

c) A recursive function that, given a list of strings, returns the string formed by their initial letters. E.g. if the parameter is `["Eleven", "Men", "in", "Flight"]` the result is `"EmiF"`.

[6]

d) A tail recursive function that computes the factorial, *n!*, of a given parameter, *n*. You may assume that *n* will be a non-negative integer.

[5]

Question 6

a) Write a recursive Prolog predicate `listelement(L, E)` that determines whether list `L` contains element `E` (but do not use Prolog's built-in `member` predicate).

[6]

b) Assume that information about all students at a university have already been entered into Prolog, under a predicate named `student(Name, Courses)`, where `Name` is the student's name and `Courses` is a list of names of courses that he follows.

i. Write a query to find the total number of courses followed by a student named `joe`.

[2]

ii. Define a predicate to determine the total number of the university's students.

[6]

iii. Define a predicate `busy(People)` that will bind `People` to a list of names of students who are all following more than 5 courses.

[6]