# UNIVERSITY OF SWAZILAND

# SUPPLEMENTARY EXAMINATION 2005

**Title of paper: PROGRAMMING LANGUAGES**

**Course number: CS343**

**Time allowed: Three (3) hours**

**Instructions: Answer any five (5) of the seven (7) questions.**

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

8/

**Question 1**

(a) Describe in detail the role of the program stack in the implementation of parameter passing, local variables and recursion.

[14]

(b) Draw a parse tree for the following expression:

►A▲►►B▼C▲D▲E▼F◄

Assume that A, B, C, D, E and F are terminals, and that the 4 triangular symbols are symbols denoting operators that possess the following properties:

| | Precedence | Arity | Fixity | Associativity |
|---|---|---|---|---|
| ◄ | 0 (highest) | 1 | Postfix | Left |
| ► | 1 | 1 | Prefix | Right |
| ▲ | 2 | 2 | Infix | Left |
| ▼ | 3 (lowest) | 2 | Infix | Right |

[6]

## Question 2

(a) Present the arguments made by structured programming advocates against the following:

(i) Global variables.

(ii) Goto.

[4]

(b) (i) Define the term *abstract data type* (ADT.)

(ii) Explain the advantages of typed programming languages over untyped languages.

[7]

(c) Explain why each of the following language features is considered to be important in supporting programming-in-the-large:

(i) Modularity.

(ii) Interface/implementation separation.

(iii) Separate compilation.

[9]

## Question 3

(a) Explain the statement: "An object possesses state, behaviour and identity."

[4]

(b) Explain the *repeated inheritance problem* in languages that support multiple inheritance.

[4]

(c) Explain how C++ supports any 3 forms of routine polymorphism. Illustrate each case with a short fragment of code.

[12]

## Question 4

(a) Define the following terms as they relate to functional programming:

(i) Type signature.

(ii) Infinite data structure.

(iii) Currying.

(iv) Pattern matching.

[8]

(b) (i) Define the structure of expressions in the λ–calculus.

[6]

(ii) Show how the following λ–calculus expression is reduced to normal form:

$$((\lambda x.\ ((\lambda y.\ x+y)\ 3))\ 2)$$

[6]

## Question 5

(a) Give the algorithm for unification of Prolog terms.

[10]

(b) Assuming that the program given further below has already been entered into Prolog, draw the search tree for the following query:

```
% This is the query:
c(X).

% Program follows:
a(1, 2, 3).
a(3, 4, 5).
b(2, 3).
c(1).
c(X) :- a(_, X, _), b(X, _).
```

[10]

## Question 6

(a) Rewrite the following infix Haskell expression in *prefix* form:

```
1+2/3
```

[2]

(b) What is meant by the following Haskell type signature?

```
f :: Int -> [String] -> [(Int, String)]
```

[4]

(c) Assume that a list named `persons`, containing names and ages of people, has been defined in Haskell in the following form:

```
persons = [("Joe", 12), ("Sam", 11), etc.]
```

(i) Define a function that returns the smallest age found in the `persons` list.

[3]

(ii) Define a function that returns the sum of ages of all members of the `persons` list.

[4]

(iii) Define a function that returns the number of members of the `persons` list representing only people aged between 10 and 15, inclusive

[7]

**Question 7**

(a) Define a recursive Prolog predicate numzeros(Nums, Zeros) that succeeds when Zeros is the number of zeros inside the Nums list argument (which has already been bound to a list of numbers.)

[4]

(b) Define a recursive Prolog predicate maximum(Nums, Max) that succeeds when the List argument (which has already been bound to a list of numbers) has Max as its greatest element. If the list is empty, the predicate should bind Max to zero.

[5]

(c) Define a recursive Prolog predicate final(List, Last) that succeeds when the List argument (which has already been bound to a list) has Last as its final element. If the list is empty, the predicate should bind Last to zero.

[5]

(d) Define a recursive Prolog predicate range(First, Last, Result) that succeeds when First and Last are integers and Result is bound to a list of integers of the form [First, First+1, First+2, ..., Last-1, Last] (i.e. all integers between First and Last, inclusive.) If First is greater than Last, Result should be bound to the empty list.

[6]