UNIVERSITY OF SWAZILAND

SUPPLEMENTARY 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) Explain the main advantages of high level programming languages.

[10]

b) Contrast between the operational, axiomatic and denotational approaches to specifying language semantics. In addition, specify the semantics of the assignment operator using each method.

[10]

Question 2

Give an overview of the kinds of user defined data types. Provide fragments of source code in C++ and/or Pascal to show how each kind of type is defined.

[20]

Question 3

a) Give two examples of unstructured programming constructs in Pascal.

[2]

b) Describe any 2 of the main prescriptions ('good practices') of structured programming.

[6]

c) Explain the meaning of inheritance and dynamic dispatch in object oriented languages. Furthermore, explain how they combine to provide inclusion polymorphism.

[8]

d) Explain the problem of repeated inheritance.

[4]

Question 4

a) Contrast between the imperative and declarative approaches to language design. In addition, explain the main advantages of each.

[7]

b) Explain the following terms in relation to functional programming:

[10]

- Referential transparency.
- Higher order function.
- Lazy evaluation.
- Type inference.
- Pattern matching.

c) Give an example of an extra-logical feature of Prolog, and explain why it is included in the language.

[3]

Question 5

Define the following functions in Haskell. In addition, *write the type signature of each function.*

a) A function that, given two lists of identical length consisting of floating-point numbers, returns a list whose n-th element is the product of the n-th elements of the given lists. E.g. if the parameters are `[5, 2.2, -3.3]` and `[-1.1, 1, -1]`, then the result is `[-5.5, 2.2, 3.3]`.

[4]

b) A function that, given a string, returns the number of upper-case characters in the string.

[8]

c) A tail-recursive version of the following function that counts the number of elements in a given list (but do not use Haskell's built-in `length` function):

[8]

```
count lst =
  if lst == [] then 0
  else 1 + count (tail lst)
```

Question 6

a) What, in general, would the user have to enter to cause the following Prolog query to succeed?

[3]

```
read([_|T]), length(T, L), L>2.
```

b) Write a recursive Prolog predicate `listlen(L, Num)` that binds `Num` to the total number of elements in the given list `L` (but do not use Prolog's built-in `length` predicate).

[7]

c) Assume that information about all students at a university has already been entered into Prolog, under a predicate named `student(Name, ID)`, where `Name` is the student's name, `ID` is his unique 6-digit identity number (in range 100000 to 999999, inclusive).

i. Define a predicate `idrange(Number)` that succeeds when the given `Number` is in the range of valid ID numbers.

[2]

ii. Write a query to find whether any two students are both named `joe`.

[3]

iii. Define a predicate `names(N)` that binds `N` to a list of all names of students.

[5]