UNIVERSITY OF SWAZILAND

Faculty of Science

Department of Computer Science

MAIN EXAMINATION 2008

Title of paper: **PROGRAMMING LANGUAGES**

Course number: **CS343**

Time allowed: 3 hours

Special requirements: Computer with Haskell and Prolog interpreters installed.

Instructions:

- Section A (Theory, 60 marks) consists of questions 1 to 5. Answer any 4 of them in the examination folder provided.
- Section B (Practical, 40 marks) consists of questions 6 and 7. Answer both questions. Save each answer in a separate file named 'Q<*num*>-<*id*>' where *num* is the question number an *id* is your student ID (e.g. Q6-654321 and Q7-654321).

THIS EXAMINATION PAPER SHOULD NOT BE OPENED UNTIL PERMISSION HAS BEEN GRANTED BY THE INVIGILATOR

Section A, Theory (60 marks)

Answer any 4 questions in the answer folder provided

Question 1

a) Explain the meaning of *semantic gap*. In addition, explain the difference in semantic gap between *high level* and *low level languages.*

[5]

b) Contrast between the way *compilers* and *interpreters* perform translation.

[2]

c) In a certain language, the 3 symbols ◊, □ and ∘ denote operators. Assume that the expression:

1 ∘ (2 ◊) □ 3 ∘ 4 □ 5 ◊

...when fully parenthesized, becomes:

(1 ∘ ((2 ◊) □ 3)) ∘ ((4 □ 5) ◊)

Explain what, if anything, can be deduced about the *arity, fixity, precedence* and *associativity* of these operators.

[8]

Question 2

a) Describe any 2 kinds of user defined data types. Illustrate each kind with code in an actual programming language showing how the data type is defined.

[4]

b) Contrast between the following pairs of terms:

i. Typed and untyped languages.

ii. Static and dynamic typing.

[4]

c) What do you understand by the term *polymorphic operator*? In addition, explain the *conversion* and *overloading* forms of polymorphism.

[7]

Question 3

a) Describe the *goto* statement and explain why its use is discouraged.

[4]

b) Give a detailed overview of the object oriented paradigm. Your answer should include explanations of the terms *object, class, inheritance* and *dynamic dispatch*.

[11]

Question 4

a) With the help of code examples, explain the meaning of the following features of Haskell:
   i. Infinite lists.
   ii. Lazy evaluation.

[8]

b) Explain the meaning of *referential transparency* and *higher order functions*, as well as their importance in functional programming.

[7]

Question 5

a) Briefly describe the purpose of the *knowledge base* and *inference engine* in logic programming systems.

[2]

b) Distinguish between the following concepts of Prolog:
   i. = and =:= predicates.
   ii. `bagof` and `setof` predicates.
   iii. atom and list data types.

[6]

c) Read the following Prolog program and *list all unification and backtracking steps* resulting from the query: `grand(A, ann).`

```
parent(jim, ann).
parent(sue, ann).
parent(joe, tom).
parent(joe, jim).
grand(A, B) :- parent(X, B), parent(A, X).
```

[7]

Section B, Practical (40 marks)

Save answers to questions 6 and 7 in separate files

Question 6

a) Write a Haskell expression of the form:

```
let a=any number ; b=any number ; c=any number
in ...
```

that returns the difference between the highest and lowest of the 3 given numbers (`a`, `b` and `c`).

[4]

b) Write a Haskell expression of the form:

```
let str=any string
in ...
```

that returns the number the number of occurrences of the 2 lower-case characters `'x'` and `'y'` in the given string (`str`). E.g. if `str` is `"your excellency"` the expression must evaluate to 3.

[7]

c) Write a Haskell expression of the form:

```
let n=any positive integer
in ...
```

that returns a list of integers starting with `n` and ending with 0, and where each item is the quotient (integer part) of dividing its predecessor by 2. E.g. if `n` is 57, the expression must evaluate to `[57, 28, 14, 7, 3, 1, 0]`. You are permitted to assume that `n` will never be zero or negative.

[7]

d) Write a Haskell expression of the form:

```
let chars=any string ;
    counts=any list of positive integers
in ...
```

that returns a list of strings. Specifically, the *i*-th item of the returned list must consist of the *i*-th character of `chars` repeated a number of times equal to the *i*-th item of `counts`. E.g. if `chars` is `"Hoho!"` and `counts` is `[2,1,2,1,3]`, the expression must evaluate to `["HH", "o", "hh", "o", "!!!"]`. You are permitted to assume that `chars` and `counts` will always be of equal length, and that `counts` will never contain zero or negative numbers.

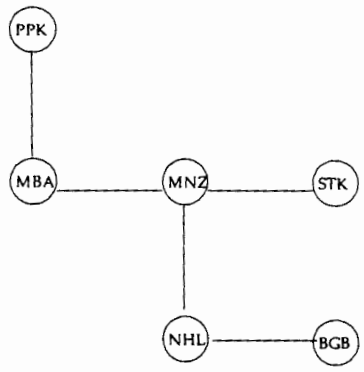
[7]

Question 7

a) The adjoining map shows 5 roads connecting 6 towns in Swaziland. Represent information about the roads by writing 5 Prolog facts, each of the form:

```
road(Town1, Town2)
```

where `Town1` and `Town2` are the two towns connected by the road.



[2]

b) Define a Prolog rule of the form:

```
link(Town1, Town2) :- ...
```

that succeeds when `Town1` and `Town2` are directly linked by a single road. E.g. based on the above map, the queries `link(mnz, mba)` and `link(mba, mnz)` must succeed, while `link(mnz, ppk)` must fail.

[3]

c) Define a Prolog rule of the form:

```
nearby(Town, Num) :- ...
```

that succeeds when `Num` is the number of towns directly linked (as defined in part b) to the given `Town`. E.g. based on the above map, the query `nearby(mnz, 3)` must succeed.

[6]

d) Define a Prolog rule of the form:

```
remote(Town) :- ...
```

that succeeds when `Town` has fewer than 3 nearby towns (as defined in part c), with the special exception of `mba`. E.g. based on the above map, only the queries `remote(mnz)` and `remote(mba)` must fail.

[4]