UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B. Sc. Examination 2011

COMPUTER SCIENCE

IS51015A Computer Science 1

Duration: 1 hour 30 minutes

Date and time:

There are three questions in this paper. You should attempt them all. The total number of marks for this paper is 100. The marks for each part of a question are indicated at the end of the part in [.] brackets.

No calculators should be used.

THIS EXAMINATION PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

QUESTION 1

(a) For each of the following types, give one example of a value of that type:

(i) `num X num`

(ii) `num X bool`

(iii) `num X num X num`

(iv) `char X num`

[4 Marks]

(b) Give the value of each of the following boolean expressions:

(i) `true and false;`

(ii) `false or true;`

(iii) `not(false or true);`

(iv) `not(not(false) and true);`

[4 Marks]

(c) Given the following truth table:

p	q	p implies q
T	T	T
T	F	F
F	T	T
F	F	T

define `implies: bool X bool -> bool;` in Hope, using `or` and `not`, by completing the right hand side of the following definition:

```
implies(p,q) <=
```

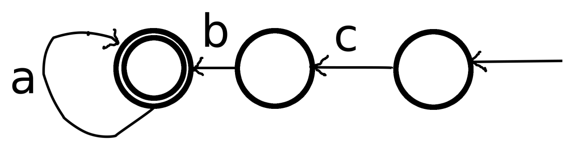
[4 Marks]

(d) Make a truth table for the function `f` given by:

```
f:bool X bool -> bool;
f(p,q) <= p and (not(q));
```

[4 Marks]

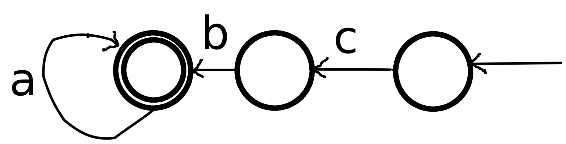
(e) Write a regular expression corresponding to the finite state machine below.



(The state containing two circles represents a 'stop' state).

[4 Marks]

(f) What is the language accepted by the finite state machine below.



(The state containing two circles represents a 'stop' state).

[4 Marks]

(g) Draw a finite state machine for the regular expression $(a|b)^*c$

[4 Marks]

(h) Give a regular expression whose language is recognised by the function `f` below:

```
f: list(char) -> bool;
g:  list(char) -> bool;

f(nil) <= false;
f(x::l)<= if x='a' or x='b'
          then g(l)
          else false;

g(nil) <= false;
g(x::nil) <= x='c';
g(x::(y::l)) <= if x='c'
                then g(y::l)
                else false;
```

[5 Marks]

QUESTION 2

(a) Define a function `max(m,n)`

```
max:num X num -> num;
```

such that `max(m,n)` returns the larger of `m` and `n`. For example, `max(2,4)` returns `4:num`.

[4 Marks]

(b) Using `max`, above, define a function `maxOf3` which returns the maximum of three numbers (you must not use an `if`).

[4 Marks]

(c) Given the functions:

```
head: list(alpha) -> alpha;
head(x::m) <= x;

tail: list(alpha) -> list(alpha);
tail(x::m) <= m;
```

give the value and the type of each of the following expressions:

(i) `head([1]);`

(ii) `tail([1]);`

(iii) `head(tail([1,3,2]));`

(iv) `tail(tail([1,3,2]));`

[4 Marks]

(d) Given the function:

```
f: list(alpha) -> num;
f(nil) <= 0
f(x::m) <= 1+ f(m);
```

give the value and the type of each of the following expressions:

(i) `f([79]);`

(ii) `f([1,2,3,1]);`

(iii) `f(tail([1]));`

(iv) `f(tail(tail([1,3,2])));`

[4 Marks]

(e) Given the two functions:

```
firstfew: num X list(alpha) -> list(alpha);
firstfew(0,k) <= nil;
firstfew(n+1,x::m) <= x:: firstfew(n,m);

lastfew: num X list(alpha) -> list(alpha);
lastfew(0,k) <= k;
lastfew(n+1,x::m) <= lastfew(n,m);
```

What is the value and type of `firstfew(3,lastfew(3,[1,2,3,4,5,6,7,8]));`?

[4 Marks]

(f) Write a function for adding up all the numbers in a list of numbers.

[4 Marks]

(g) Write a function `elementAt: num X list(alpha) -> alpha` such that `elementAt(n,k)` returns the element at position `n` (starting from 0) in the list `k`.

[4 Marks]

(h) Write a function which takes a list and returns its 'middle' element (If the list has an even number of elements, then make a reasonable choice for the middle element). You may assume the `length` function has already been defined. (You may use `elementAt` above.)

[5 Marks]

QUESTION 3

(a) Give the value and the type of each of the following expressions:

(i) `1 & empty;`

(ii) `([] & empty) U ([2] & empty);`

(iii) `'a' isin ('b' & empty);`

(iv) `[1,2] isin ([1,2] & empty);`

[4 Marks]

(b) Briefly describe the differences between sets and lists.

[4 Marks]

(c) Describe what the function `gg`, below, does.

```
gg: set(alpha) -> num;
gg(S) <=  if S = empty
          then 0
          else let (a,T) == choose(S)
               in 1 + gg(T);
```

[4 Marks]

(d) Describe what the function `ff`, below, does.

```
ff: set(alpha) X set(alpha) -> set(alpha);
ff(S1,S2) <=  if S1 = empty
              then empty
              else let (a,S3) == choose(S1)
                   in if a isin S2
                      then a & ff(S3,S2)
                      else ff(S3,S2);
```

[4 Marks]

(e) The set difference between X and Y is the set of elements that are in X but not in Y. Define the function:

```
setDifference: set(alpha) X set(alpha) -> set(alpha);
```

Hint: it will be similar to the function `ff`, above.

[4 Marks]

(f) A directed graph whose nodes (vertices) are of type alpha can be represented as

```
type graph(alpha) == set(alpha X alpha);
```

where each pair (x, y) in the set represents an edge from x to y.

Describe what the following function, `f`, does.

```
f: graph(alpha) -> set(alpha);
f(G) <= if G=empty
        then empty
        else let ((a,b),G1) == choose(G)
             in (a & (b & empty)) U f(G1);
```

[4 Marks]

(g) The *out-degree* of a vertex v in a graph g is the number of edges of g emerging from v. Write a function:

```
outdegree: alpha X graph(alpha) -> num;
```

such that `outdegree(v,g)` returns the out-degree of vertex `v` in the graph `g`.

[5 Marks]

(h) Write a function, `nexts`, which given a vertex v and a graph g, finds the set of vertices that are at the end of an out-going edge from v.

[5 Marks]