P. S.

Family Name:	
Other Names:	
Signature:	
Student Number:	

The University Of New South Wales

Final Exam

November 1998

COMP1011

Computing 1A

Time allowed: **3 hours**
Total number of questions: **28**
Total number of marks: **100**
No examination materials permitted.
Calculators may not be used.
Questions are not worth equal marks.
There is one mark awarded for following the examination instructions.

Examiner's Use Only										
	Inst	A	B	23	23	25	26	27	28	Total

UNSW

>999843451

Part A: Multiple Choice

Answer the questions in this part by filling in entries on the multiple-choice sheet supplied.

Note that each question has *five* alternatives. Once you have chosen an alternative, fill in the multiple-choice sheet by giving the letter (in square brackets e.g. "[B]") which corresponds to that alternative. Be careful to fill each answer in on the correct row on the multiple-choice sheet (i.e. the row corresponding to the question number).

For each correct answer you **earn 2 marks**, for each incorrect answer you will **lose 0.5 marks**

Question 1

Which of the following is the best name for a module which generates random numbers?

[A] `b7`
[B] `randomGenerator`
[C] `ThisIsARandomNumberModule`
[D] `Random`
[E] `leonardo_de_caprio`.

Question 2

Which of the following Haskell patterns will match only lists of length two?

[A] `[a,b]`
[B] `[a,b,c]`
[C] `[_,_,[]]`
[D] `(a:b)`
[E] More than one of the above.

Question 3

User richardb has a file `/home/richardb/public_html/example.html`. What address can you use to view this file with a web browser?

[A] `http://www.cse.unsw.edu.au/home/public_html/richardb/example.html`
[B] `http://www.cse.unsw.edu.au/~richardb/example.html`
[C] `http://www.cse.unsw.edu.au/richardb/public_html/example.html`
[D] `http://www.cse.unsw.edu.au/~richardb/public_html/example.html`
[E] More than one of the above.

Question 4

User gwen has created a CGI program. The shell script file is `ROUS` and the Haskell script file is `Rous.hs`. She has placed the files in her `cgi-bin` directory.

What address can you use to execute this script with the data "Humperdink" and display the output as an html document?

```
[A] http://cgi-sol86.cse.unsw.edu.au/cgiwrap/cs1011cgi/~gwen/ROUS?Humperdink
[B] http://cgi-sol86.cse.unsw.edu.au/cgiwrap/cs1011cgi/gwen/ROUS?Humperdink
[C] http://cgi-sol86.cse.unsw.edu.au/cgi-bin/cgiwrap/gwen/ROUS?Humperdink
[D] http://cgi-sol86.cse.unsw.edu.au/cgi-bin/cgiwrap/~gwen/ROUS?Humperdink
[E] http://www.cse.unsw.edu.au/~gwen/cgi-bin/Rous?Humperdink
```

Question 5

The file `test.txt` contains only spaces, alphabetic characters, numeric digits, and full stops. Which of the following could you **NOT** determine using a one line grep command using only the options of in ass2: cvE, and the special characters of ass2: `*.^$\\[]+?{,\}()|`, and using a pattern of less than 50 characters in length?

[A] list all lines containing no vowels
[B] list all lines in the file containing an adjacent pair of repeated characters
(such as "ee" or "33")
[C] list all lines entirely made up of alternating sequences of vowels and consinants
(such as "EtaDonut" or "ZeXeCeVa")
[D] Two or more of the above cannot be determined.
[E] They can all be determined.

Question 6

Consider the following type definitions:

```
(1) type NewType = ADT
(2) data NewType = Saturday | Sunday
(3) data NewType = Board (Int,Int)
```

Which is a suitable definition for an ADT?

```
[A] (1)
[B] (1) and (3) but not (2)
[C] (1) and (2) but not (3)
[D] (1) and (2) and (3)
[E] (2) and (3) but not (1)
```

Questions 7, 8, 9, and 10 refer to the following Haskell definitions:

```
f 0 = [0]
f 1 = [1]
f x = x^2 : f (x-2)

g [] y     = y
g (x:xs) y = g xs (x * y)

h a d = take a (map d [(d a), (d (a+1)) ..])

k x y
 | x > 0       = x*y + k (x-1) (y+1)
 | otherwise   = 0
```

Question 7

What is the result of evaluating `length (f 7)`?

[A] 0
[B] 3
[C] 4
[D] 5
[E] An error message

Question 8

What is the result of evaluating `g (f 5) 3`?

[A] 0
[B] 675
[C] 33075
[D] 50505
[E] An error message

Question 9

What is the result of evaluating `:t h`?

```
[A] h :: Int -> Int -> (Int -> a)
[B] h :: Int -> (Int -> a) -> [Int]
[C] h :: Int -> a -> [Int]
[D] h :: (Int -> a) -> Int -> [Int]
[E] h :: Int -> (Int -> Int) -> [Int]
```

Question 10

What is the result of evaluating `head (h 6 (k 2))`?

[A] 0
[B] 12
[C] 19
[D] 58
[E] An error message

Question 11

Consider the list comprehension `[ x | x <- list, x < y ]`. If you wished to do this using functions instead, which of the following builtin functions would be most useful?

[A] `sum`
[B] `sort`
[C] `map`
[D] `filter`
[E] `repeat`

Question 12

For which of the following types does it not make sense to derive `Ord`?

```
[A] data Direction = North | East | South | West
[B] data Size = Biggest | Large | Medium | Small | Smallest
[C] data WeekDay = Monday | Tuesday | Wednesday | Thursday | Friday
[D] data Cardinal = Finite Integer
                  | Infinite
```
[E] It doesn't make sense for any of the type definitions above

Part B: Short Answer Questions

Answer the questions in the spaces provided on this exam sheet.
DO NOT answer these questions in an exam booklet.

Write your answers clearly. Keep your answers neat and very brief. Messy or long answers will not be marked.

Each question is worth **2 marks.**

Question 13

Student s5554567 is using her account at cse. She is currently in the directory /home/s5554567/labs.

The cd command is used to change directories. s5554567 wants to change to her home directory. Write down 3 possible cd commands that she could use:

Answer: ____

____

____

Question 14

Give two different Unix commands which will create a backup copy of the file `mydata.save` in a file named `mydata.old`. Your answers must each be single line Unix shell commands. They must create the backup copy without further input from the user. Hence your answer can not involve interactive programs such as **nedit** or **hugs**. The programs (not just the arguments) must be different.

Answer (a): ____

Answer (b): ____

Question 15

What are two of the most common recursion errors?

Answer (a): ____

Answer (b): ____

Question 16

Write a list comprehension which returns all the numbers between 0 and n which are square numbers. For example 16 is a square number (4^2), but 23 is not.

Answer: ____

The next two questions refer to the following information:
A Unix program `showlabs` *prints out the lab allocations for all students. Each line is formatted like this:*

```
s2109876  fri17-flute
```

Question 17

The unix command `showlabs > labs.txt` has been used to save the output of `showlabs` in the file `labs.txt`. Give a single line unix command to extract your own lab allocation information from `labs.txt`.

Answer: ____

Question 18

Now give a one line command to get this information directly from `showlabs`, without using `labs.txt`.

Answer: ____

Question 19

Give a unix command to find all lines in the file `Gwen.txt` which contain entire sentences without punctuation (other than the closing full stop).

You may assume that sentences start with capital letters, finish with full stops, and that the only capital letters in the file are those at the start of sentences. You may also assume that the file consistes only of letters, spaces, and punctuation characters.

For example the only line in:

```
Gwen isn't a boy.  She is a
girl.  She has golden hair and
a glowing smile.  She is perfect.
```

which satisfies these conditions is

```
a glowing smile.  She is perfect.
```

Answer: ____

Question 20

You are designing a web site for a client. You have already designed a similar site for a previous client. You would like to re-use programs and material from the earlier project in the current project.

There are a number of legal issues involved in cases such as this. List three.

Answer: ____

____

____

Question 21

The `labs` directory contains the following:

```
Lab04 lab02 lab04
file.txt lab03 lab05
```

`file.txt` is a file. Everything else is a directory with contents as follows:

```
% ls Lab04
MyLab.hs

% ls lab02
Dictionary.hs Lab02.hs
```

Assume you are in the `labs` directory. Write a series of (no more than 7) commands to do the following:

```
Put file.txt into the lab05 directory
Copy Dictionary.hs from lab02 into lab05
Move MyLab.hs from Lab04 into lab04, and rename it Lab04.hs
Delete the directory Lab04
```

Answer: ____

Question 22

Briefly describe what is wrong with the following function:

```
myRead :: String -> Int
-- converts string representation of an Int to the numeric
-- representation of the number.
myRead [] = 0
myRead (x:xs)
 | isDigit x = (10 * myRead xs) + (ord x - ord '0')
 | otherwise = error "myRead: Non numeric string"
```

Answer: ____

Part C

Start this part in the first script booklet. Start each question on a new page.

Make your answers as clear and easy to understand as possible. Provide type definitions and brief comments where necessary. Confusing or illegible solutions will lose marks.

If you need to use more than one script book, ensure that you fill in all of the details on the front of each script book.

If you do not wish your answer for a question to be marked record 1 mark for that question on the front of the script book. If you do this your answer for that question will **not** be marked.

In this part you may **not** import or use any library modules.

Question 23

(5 marks)

Write a Haskell function, *isPrime*, which takes as input an integer greater than one, and which returns True if the integer is prime and False otherwise.

You may assume that the input is greater than one.

Question 24

(10 marks)

A sublist is a list with some or no elements deleted. For example the sublists of [2,4,1] are

```
[2,4,1]
[2,4]
[2,1]
[4,1]
[2]
[4]
[1]
[]
```

Write a Haskell function, *subLists*, which takes as input a list, and returns a list of all its sublists. For example:

```
Exam> subLists "Sub"
["", "b", "u", "ub", "S", "Sb", "Su", "Sub"]
Exam> subLists [8,6]
[[], [6], [8], [8, 6]]
```

Part D

Start this part in the first script booklet. Start each question on a new page.

Make your answers as clear and easy to understand as possible. Provide type definitions and brief comments where necessary. Confusing or illegible solutions will lose marks.

If you need to use more than one script book, ensure that you fill in all of the details on the front of each script book.

If you do not wish your answer for a question to be marked record 1 mark for that question on the front of the script book. If you do this your answer for that question will **not** be marked.

In this part you may **not** import or use any library modules.

Question 25

(5 marks)

Write a Haskell function, *increasing*, which takes as input a list of intgers, and which returns True if and only if the differences between adjacent pairs of integers are increasing.

For example, the differences of the list [1,2,4,7] are 1,2,3 which are increasing, whereas the differences of [1,3,5,8] are 2,2,3, which are not increasing.

Here are some examples of how the function is to behave:

```
Exam> increasing [1,2,4,7]
True
Exam> increasing [1,3,5,8]
False
Exam> increasing [3,6,1]
False
Exam> increasing [1,10,100,1000]
True
```

You may assume that the list has at least two elements.

Question 26

(10 marks)

The examples in this part make use of the following functions:

```
myToLower :: Char -> Char
myToLower inChar
  | elem inChar ['A' .. 'Z'] = chr ((ord inChar) + (ord 'a') - (ord 'A'))
  | otherwise                = inChar

dumbToLower :: Char -> Char
dumbToLower inChar
  | elem inChar ['A' .. 'Y'] = chr ((ord inChar) + (ord 'a') - (ord 'A'))
  | otherwise                = inChar
```

Write a Haskell function,

```
fn_equal :: (Enum a, Bounded a, Eq b) => (a -> b) -> (a -> b) -> Bool
```

which compares two functions and returns true if and only if the functions are equal.

For example:

```
Exam> fn_equal myToLower toLower
True
Exam> fn_equal dumbToLower toLower
False
Exam> fn_equal (not) (not.not.not)
True
Exam> fn_equal (not.not) (not.not.not)
False
Exam> fn_equal (*2) (+2)
False
Exam> fn_equal ((*2) . (‘div‘ 2))  (*1)
False
```

Part E

Start this part in a new script booklet. Start each question on a new page.

In this part partial solutions (i.e. attempts worth less than 50% of the marks for a question) will score no marks.

Make your answers as clear and easy to understand as possible. Provide type definitions and brief comments where necessary. Your solutions must be clear, elegant and easy to understand. In this part confusing or difficult to understand solutions will score no marks.

If you need to use more than one script book, ensure that you fill in all of the details on the front of each script book.

If you do not wish your answer for a question to be marked record 1 mark for that question on the front of the script book. If you do this your answer for that question will **not** be marked.

In this part you **may** import and use any of the standard Hugs 1.4 library modules

Question 27

(12 marks)

Write a function `grop :: [Char] -> [Char] -> [[Char]]` which takes as input a pattern and a data string, and returns a list of all the distinct fragments of the data string which match the pattern.

The only character which has a special meaning in the pattern is '*', which matches 0 or more characters. Note that this differs from the meaning of '*' in *grep*. The '*' in *grop* matches any characters, not just repeated instances of the character preceeding the '*'.

For example:

```
Exam> grop "at" "ratat"
["at"]
Exam> grop "*" "Dim"
["", "D", "Di", "Dim", "i", "im", "m"]
Exam> grop "i*" "Dim"
["i", "im"]
Exam> grop "*i" "Dim"
["Di", "i"]
Exam> grop "*i*" "Dim"
["Di", "Dim", "i", "im"]
Exam> grop "1*9" "191291abc9"
["19", "19129", "191291abc9", "129", "1291abc9", "1abc9"]
```

Question 28

(13 marks)

Your task is to calculate the shortest sequence of moves to take a knight from a specified square on a chessboard to another specified square.

A chessboard is an 8×8 grid. The squares are labelled by coordinate pairs, as in assignment III. A knight makes an L-shaped move. It moves either two spaces horizontally followed by one square vertically, or two squares vertically followed by one horizontally.

For example, a knight on (3,4) can move to any of the eight squares: (1, 3), (1, 5), (2, 2), (2, 6), (4, 2), (4, 6), (5, 3), or (5, 5).

Moves off the board are not allowed, so a knight at (1,4) has only four alternatives for its next move: (2, 2), (2, 6), (3, 3), or (3, 5).

The following fragment of a Haskell module generates the infinite tree of possible sequences of knight moves from a given square:

```
type Pos  = (Int,Int)
type Path = [Pos]
data Tree = Node Pos [Tree]

next_squares:: Pos -> [Pos]
next_squares (x,y) = [(x+x_offset,y+y_offset)|
                      x_offset <- offsets,
                      y_offset <- offsets,
                      (abs x_offset) /= (abs y_offset),
                      elem (x+x_offset) [1..8],
                      elem (y+y_offset) [1..8]]
   where
      offsets = [(-2),(-1),1,2]

moves :: Pos -> Tree
moves from = Node from (map moves (next_squares from))
```

Write a Haskell function `path :: Pos -> Pos -> Path` which, given the coordinates of a start square and the coordinates of a finish square, returns the shortest sequence of knight moves from the start square to the finish square.

If there is more than one sequence of this length then the function may return any of them.

Constraint: Your funtion is to extract the seqence of moves from the output of the function `moves` above. You will need to make use of lazy evaluation in order to return a result in a finite period of time.

You may assume that the above fragment is already entered, you need not rewrite it in your answer.