Answer Question 1 and any TWO other questions.

You may find the following definitions useful throughout this paper:

```
id x = x

const x y = x

lay []    = []
lay (a:x) = a ++ "\n" ++ (lay x)

foldr f def [] = def
foldr f def (front : rest)
        = f front (foldr f def rest)

map f []     = []
map f (x:xs) = (f x) : (map f xs)

takewhile f []     = []
takewhile f (x:xs) = x : (takewhile f xs), if f x
                   = [], otherwise
```

1. (a) Write a function (with types, comments and auxiliary functions) that takes a list of numbers as input and returns a histogram, with vertical columns of stars rising to heights equal to the values of the numbers given. For example, if your function is called "hist" then a sample interaction with the Miranda system would be as follows:

```
Miranda   hist [3,1,5]
    *
    *
*   *
*   *
* * *
```

[13]

(b) Write a function (with types, comments and auxiliary functions) which takes a list of names and returns a list of every possible committee that could be formed from amongst those names. For example, if your function is called "cttee" then a sample interaction with the Miranda system would be as follows:

```
Miranda  cttee ["Anne", "Brijesh", "Carol"]

[["Anne"],["Brijesh"],["Carol"],["Anne","Brijesh"],
 ["Anne","Carol"], ["Brijesh","Carol"],
 ["Anne","Brijesh","Carol"]]
```

Note that the order of the committees and of the names in them is not important.

[10]

[Question 1 cont. over page]
[TURN OVER]

[Question 1 cont.]

(c) Evaluate each of the following Miranda expressions. If you think that an expression gives an error, say why (if there is more than one error in an expression, explain all of them):

(i)
```
[ ] : [ [ ] ] : [ [ [ ] ] ]
```

(ii)
```
[ [ ] ] : [ ] : [ [ [ ] ] ]
```

(iii)
```
[ [ [ ] ] ] : [ ] : [ [ ] ]
```

(iv)
```
[ ] ++ [ ] ++ [ ]
```

(v)
```
exp1 13
where
exp1 x = "False", if (3 < x < 27)
       = False,   otherwise
```

(vi)
```
exp2 13
where
exp2 = 3, if True
     = 5, otherwise
```

(vii)
```
exp3 5 9 27
where
exp3 x = (id . (+) . (const x)), if False
       = g, otherwise
g a b  = a + b
```

[9]

(d) Explain, with examples, what is meant by the following terms:

algebraic type

partial application

structural induction

[8]
[Total 40]

[CONTINUED]

2. Given the following extract from a Miranda literate script:

```
Insertion sort program.
Elements of an unsorted list are inserted,
one at a time, into a sorted list.

> isort :: [*] -> [*]
> isort [] = []
> isort (front : rest)
>       = insert (front, isort rest)
>         where
>           insert (x, []) = [x]
>           insert (x, y : ys) = x : y : ys, if x <= y
>                              = y : insert (x,ys), otherwise
```

(a) Give Miranda's response for each of the following expressions assuming it is typed at the Miranda prompt:

```
isort "" ::

isort [isort,isort] ::

insert (2,[])

isort [3..1]

(hd . isort) ["deck","of","cards"]

isort [isort,isort]
```

[6]

(b) Briefly explain the advantages of using *Higher Order* functions. Illustrate your answer by generalizing `isort` to create a new *curried* function `gsort`, which can sort a list of any type in any user-provided ordering. Give both the type declaration and function definition of `gsort`.

[8]

(c) With reference to `isort` and `gsort` discuss the advantages of *polymorphic typing* and *currying*.

[8]

(d) Provide specialized versions of `gsort`:

(i) To sort Miranda character strings in descending order.

(ii) To sort the sum of a list of 2-tuples in ascending order. For this function (2,3) is considered less than (1,7), because (2+3) is less than (1+7).

[8]
[Total 30]

[TURN OVER]

3. Discuss the similarities and differences of C++ and Miranda, paying particular attention to the following:

   — Abstraction and encapsulation

   — Dynamic data structures

   — User input and output [Total 30]

4. The game of noughts and crosses (also known as tic-tac-toe) has a three-by-three board and two players. One player has counters that are shaped like a cross and the other player's counters are shaped like a circle (a nought). This game can also be played with paper and pen, with the two players drawing noughts and crosses instead of using counters. Here is how the game works:

   A three-by-three board has nine squares. The two players decide who is to start, and that person places a counter on one of the empty squares (the squares are all empty to start). Then the second player places a counter on any square that is still empty. The winner is the player who succeeds in placing counters so that he or she has three of his or her counters in a straight line, in any direction (including the diagonals).

   (a) Provide an algebraic type to represent the state of a square and an abstype to represent the game board. Your abstype should provide interface functions as follows:

```
emptyboard

addnought

addcross

checkmove

checkwinner

showboard
```

   You should decide for yourself how these functions should work and you should provide full code (including type expressions and comments) for them. [20]

   (b) Provide the top-level user interface code that allows the user to enter coordinates (alternately for player1, player2, player1 etc.) and then calls the appropriate interface functions. Your code should stop the game when a winner has been detected, or when 9 moves have been made (in which case the result is a draw). Your code should display the state of the board after each move.

   Note: you do **not** need to write code to decide which move to make next. In this program, the computer is **not** one of the players.

   [10]
   [Total 30]

[CONTINUED]

5. (a) Given the following definition for the function called `bizarre`:

```
bizarre f x = f x (f x)
```

Explain why this definition would produce a type error.

[6]

(b) The following expression uses an infinite list within a list comprehension, in an attempt to generate all the cubes less than 5,000,000.

```
> smallcubes = [(x ^ 3) | x <- [1..]; (x ^ 3) < 5000000]
```

(a) Explain why `smallcubes` may never terminate.

(b) Provide a terminating definition for `smallcubes` which doesn't use a list comprehension.

[12]

(c) What is the most general type of each of the following functions:

```
fun1 (a:b) f x = ((x f b) . f) a

fun2 []       j        acc = acc
fun2 f        []       acc = acc
fun2 (x : xs) (y : ys) acc = y : (fun2 xs ys (x y acc))


mysterious [] y z
    = z
mysterious (front : rest) y z
    = (y z) & mysterious rest y ((front . y) z)
```

[12]

[Total 30]

[END OF PAPER]