Answer Question 1 and any TWO other questions.

You may find the following definitions useful throughout this paper:

```
fun id x = x

fun const x y = x

fun lay []     = ""
|   lay (a::x) = a ^ "\n" ^ (lay x)

fun foldr f def [] = def
|   foldr f def (x :: rest) = f x (foldr f def rest)

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

fun takewhile f []      = []
|   takewhile f (x::xs)
    = if (f x) then (x :: (takewhile f xs)) else []
```

1. (a) Write a function (with types, comments and auxiliary functions) that takes a list of numbers as input and returns a vertical histogram, with bars 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 SML system would be as follows:

```
-   hist [3,1,5] ;
val it = "    * \n    * \n*   * \n*   * \n* * * \n" : string
```

If this were printed to a file, it would give:

```
    *
    *
*   *
*   *
* * *
```

[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 SML system would be as follows:

```
-  cttee ["Anne", "Brijesh", "Carol"] ;
val it = [["Anne"],["Brijesh"],["Carol"],["Brijesh","Carol"],
          ["Anne","Brijesh"],["Anne","Carol"],
          ["Anne","Brijesh","Carol"]] : string list list
```

The order of the committees and the names in them is not important. [10]

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

[Question 1 cont.]

(c) Evaluate each of the following SML 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)
```
let
fun exp1 x = if (3 < x < 27)
             then "false" else false
in
exp1 13
end
```

(vi)
```
let
val exp2 = if true then 3 else 5
in
exp2 13
end
```

(vii)
```
let
fun g a b  = (a + b) : int
fun exp3 x = if false
             then (id o g o (const x))
             else g
in
exp3 5 9 27
end
```

[9]

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

datatype

partial application

structural induction

[8]
[Total 40]

[CONTINUED]

2. Given the following extract from an SML program::

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

fun isort ([] : 'a list) = ([] : 'a list)
|   isort (front :: rest)
      = let
          fun insert (x, []) = [x]
          |   insert (x, y :: ys)
                             = if (x <= y)
                               then (x :: y :: ys)
                               else (y :: insert (x,ys))
        in
        insert (front, isort rest)
        end
```

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

```
isort "" ;

insert (2,[]) ;

(hd o 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 an SML string list 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 SML, 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): alternatively, the game can be played with paper and pens. 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 a datatype to represent the state of a square and an abstype to represent the game board, with 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. Also note: your program does **not** need to be interactive. The list of moves (list of coordinates) should be passed as an argument to the main function.

   [10]
   [Total 30]

[CONTINUED]

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

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

Explain why this definition would produce a type error.

[6]

(b) Briefly explain the importance of garbage collection and give two examples of how this technique works.

[12]

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

```
fun f1 (a::b) f x = ((x f b) o f) a

fun f2 []         j        acc = acc
|   f2 f          []       acc = acc
|   f2 (x :: xs) (y :: ys) acc = y :: (f2 xs ys (x y acc))


fun mysterious [] y z
    = z
|  mysterious (front :: rest) y z
    = (y z) andalso (mysterious rest y ((front o y) z))
```

[12]

[Total 30]

[END OF PAPER]