You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator
SECTION A

1 Foundations of Computer Science

(a) Write brief notes on ML datatypes and pattern-matching in function declarations. [6 marks]

(b) A binary tree is either a leaf (containing no information) or is a branch containing a label and two subtrees (called the left and right subtrees). Write ML code for a function that takes a label and two lists of trees, returning all trees that consist of a branch with the given label, with the left subtree taken from the first list of trees and the right subtree taken from the second list of trees. [6 marks]

(c) Write ML code for a function that, given a list of distinct values, returns a list of all possible binary trees whose labels, enumerated in inorder, match that list. For example, given the list [1,2,3] your function should return (in any order) the following list of trees:

```
  1
 / \ /
2   3
  \
  3
```

[8 marks]

All ML code must be explained clearly and should be free of needless complexity.
The function \texttt{perms} returns all \(n!\) permutations of a given \(n\)-element list.

\[
\text{fun cons } x \ y = x :: y;
\]

\[
\begin{aligned}
\text{fun} \ \text{perms} [\] &= [[[]]] \\
| \ \text{perms} \ \text{xs} &= \\
& \quad \text{let fun} \ \text{perms1} ([], ys) = [] \\
& \quad \ | \ \text{perms1} (x :: xs, ys) = \\
& \quad \quad \text{map} (\text{cons} \ x) (\text{perms} (\text{rev} \ ys \ @ \ xs)) @ \\
& \quad \quad \ \text{perms1} \ (xs, x :: ys) \\
& \quad \quad \text{in} \ \text{perms1} \ (xs, []) \ \text{end;}
\end{aligned}
\]

(a) Explain the ideas behind this code, including the function \texttt{perms1} and the expression \texttt{map (cons x)}. What value is returned by \texttt{perms [1,2,3]}? [7 marks]

(b) A student modifies \texttt{perms} to use an ML type of lazy lists, where \texttt{appendq} and \texttt{mapq} are lazy list analogues of \texttt{@} and \texttt{map}.

\[
\begin{aligned}
\text{fun} \ l\text{perms} [\] &= \text{Cons} ([], \text{fn}() => \text{Nil}) \\
| \ l\text{perms} \ \text{xs} &= \\
& \quad \text{let fun} \ \text{perms1} ([], ys) = \text{Nil} \\
& \quad \quad \text{perms1} (x :: xs, ys) = \\
& \quad \quad \text{appendq} (\text{mapq} (\text{cons} \ x) (l\text{perms} (\text{rev} \ ys \ @ \ xs)), \\
& \quad \quad \ \text{perms1} \ (xs, x :: ys)) \\
& \quad \quad \text{in} \ \text{perms1} \ (xs, []) \ \text{end;}
\end{aligned}
\]

Unfortunately, \texttt{lperms} computes all \(n!\) permutations as soon as it is called. Describe how lazy lists are implemented in ML and explain why laziness is not achieved here. [5 marks]

(c) Modify the function \texttt{lperms}, without changing its type, so that it computes permutations upon demand rather than all at once. [8 marks]

All ML code must be explained clearly and should be free of needless complexity.
SECTION B

3 Discrete Mathematics I

(a) Consider the following assertions about the sets $A$, $B$, and $C$. Write them down in the language of predicate logic. Use only the constructions of predicate logic ($\forall$, $\exists$, ¬, ⇒, ∧, ∨) and the element-of symbol (∈). Do not use derived notions (∩, ∪, =, etc.).

Example: “$A$ is a subset of $B$” can be formalized as $\forall x. x \in A \implies x \in B$.

(i) The sets $A$ and $B$ are equal.

(ii) Every element of $A$ is in the set $B$ or the set $C$.

(iii) If $A$ is disjoint from $B$ then $B$ and $C$ overlap.

[6 marks]

(b) State the principle of induction over lists. Use the language of predicate logic.

[2 marks]

(c) Consider the following functions over lists of integers, written in ML syntax.

\[
\text{fun app([],ys) = ys} \\
| \text{app(x::xs,ys) = x::app(xs,ys)};
\]

\[
\text{fun rev([]) = []} \\
| \text{rev(x::xs) = app(rev(xs),x::[])};
\]

\[
\text{fun revapp([],ys) = ys} \\
| \text{revapp(x::xs,ys) = revapp(xs,x::ys)};
\]

Prove that $\forall xs. \text{revapp}(xs,[]) = \text{rev}(xs)$

Your proof should be clear but it does not need to be a structured proof. You may use the abbreviation $xs @ ys$ for $\text{app}(xs,ys)$. You may assume the following facts.

$\forall xs. xs @ [] = xs$ \hspace{1cm} $\forall xs,ys,zs. xs @ (ys @ zs) = (xs @ ys) @ zs$

Hint: first use induction to show that $\forall xs. \forall ys. \text{revapp}(xs,ys) = \text{app}(\text{rev}(xs),ys)$.

[12 marks]
4 Discrete Mathematics I

(a) Write down the introduction and elimination rules for the universal quantifier (\(\forall\)), the existential quantifier (\(\exists\)) and negation (\(\neg\)) in structured proof. [6 marks]

(b) Write down the introduction rule for implication (\(\Longrightarrow\)) in structured proof. [1 mark]

(c) Write down a structured proof of the following sentence.

\[(\forall x. \neg P(x)) \Longrightarrow \neg \exists x. P(x)\]  
[5 marks]

(d) Write down a structured proof of the following sentence. Clearly state any proof rules that you use in addition to those included in part (a) and part (b).

\[(\neg \forall x. \neg P(x)) \Longrightarrow \exists x. P(x)\]  
[8 marks]
5 Algorithms I

One of several ways to perform string matching efficiently is with a finite state automaton (FSA).

(a) Give a brief but clear explanation of the FSA string matching algorithm, its complexity and any associated data structures. [Note: pseudocode of up to 10 lines is allowed, but not required.] [4 marks]

(b) Build the FSA that will find matches of the pattern $P = \text{pepep}$ in an arbitrary string $T$ over the alphabet $\{e, o, p\}$, explaining what you do and why. [6 marks]

(c) The correctness proof of the FSA string matching algorithm involves the function $\sigma_P(x)$, which is parametric in the pattern $P$ and takes as input a string $x$. Define $\sigma_P(x)$, explaining what it returns. [1 mark]

(d) Let $A, B, C, D$ be character strings; let $|A|$ be the length of string $A$; let $+$ denote integer addition or string concatenation depending on its operands. Let $D$ be the longest suffix of $A$ that is a prefix of $B$.

For each of the following claims: either prove the claim correct, or give a counterexample that proves it is incorrect. You may draw an explanatory picture if it helps clarity.

(i) $\sigma_B(A) = D$ [3 marks]

(ii) $\sigma_B(A + C) = |D| + |C|$ [3 marks]

(iii) $|C| = 1 \Rightarrow \sigma_B(A + C) = \sigma_B(A) + 1$ [3 marks]
A palindrome is a string that, if reversed, remains the same, for example “madamimadam”. A subsequence of a string $x$ is one obtained by dropping zero or more characters from $x$ and taking the remaining ones in order: for example “tan” is a subsequence of “pentagon”. In this question you must find the longest palindrome subsequence (LPS) of a given string. [Note that the LPS may not be unique.]

(a) Explain why it is possible to apply dynamic programming to the LPS problem. Develop and explain a recursive equation for the length of the LPS. [6 marks]

(b) Develop and describe in detail, with pictures where appropriate, a bottom-up dynamic programming algorithm to solve the LPS problem. Include an explanation of how to recover the LPS from the bottom-up table you build. If you use pseudocode (not required), keep each pseudocode chunk under 10 lines and comment it clearly. Incomprehensible code will be scored as wrong. [9 marks]

(c) Derive the asymptotic worst-case running time of your algorithm. [2 marks]

(d) What else would you have to do to recover all the LPSs of a given string? [3 marks]
SECTION D

7 Floating-Point Computation

The following two functions are algorithms for exponentiation where \( x \) is a single-precision floating-point value and \( n \) is an integer,

\[
\text{fun power1}(x, n) = \begin{cases} 
1.0 & \text{if } n=0 \\
1.0 \times \text{power1}(x, n-1) & \text{otherwise}
\end{cases}
\]

\[
\text{fun power2}(x, n) = \begin{cases} 
1.0 & \text{if } n=0 \\
1.0 \times \text{power2}(x \times x, n \div 2) & \text{if } n \text{ even} \\
x \times \text{power2}(x, n-1) & \text{otherwise}
\end{cases}
\]

(a) What is, roughly, the largest value of \( n \) that can be used without overflow when \( x \) is 10.0? [1 mark]

(b) Suppose \( x \) is close to 1.0.

(i) What is the worst possible relative error to expect in the answer from \text{power1} when \( n = 100 \)? [3 marks]

(ii) Can we say anything useful about the absolute error in part (b)(i)? [1 mark]

(iii) What is the expected value of the relative error in results from \text{power1}? [1 mark]

(c) Sometimes the expected magnitude of error can be estimated as the result of a random walk.

(i) Under what conditions is this appropriate? [2 marks]

(ii) What is the random walk estimate for the relative error in part (b)(i)? [3 marks]

(d) If \( x \) is again close to 1.0, what is the worst possible relative error to expect from \text{power2} when \( n = 100 \)? [6 marks]

(e) For what range or class of \( x \) values will \text{power2} with \( n = 100 \) give a result with no error? [3 marks]
8 Object-Oriented Programming with Java

Sparse matrices are matrices whose elements are predominantly zero. This question develops a Java representation for them called `SparseMatrix`.

The code below seeks to use an `ArrayList` of `LinkedList`s to implement the concept efficiently. It defines a class `Element` to store the column number and value for an element. Each row is represented by a `LinkedList` of `Element`s with non-zero values only. Few, if any, rows are all zeros and so the `ArrayList` is used to store a `LinkedList` for every row in ascending row order.

```java
public class Element {
    public int column;
    public int value;
}

public class SparseMatrix {
    private int mRows; // Number of rows
    private int mCols; // Number of columns
    private ArrayList<LinkedList<Element>> mMatrix; // Data

    // Constructor
    public SparseMatrix(int rows, int cols) {
        mRows = rows;
        mCols = cols;
        mMatrix = new ArrayList<LinkedList<Element>>();
        for (int i = 0; i < rows; i++) {
            LinkedList<Element> row = new LinkedList<Element>();
            mMatrix.add(row);
        }
    }

    // Get value at row r and column c
    public int get(int r, int c) {
        int result = 0;
        Element e = findElement(r, c);
        if (e != null) {
            result = e.value;
        }
        return result;
    }

    // Set value at row r and column c to v
    public void set(int r, int c, int v) {
        Element e = findElement(r, c);
        if (e != null) {
            e.value = v;
        } else {
            LinkedList<Element> row = mMatrix.get(r);
            Element newElement = new Element();
            newElement.column = c;
            newElement.value = v;
            row.add(newElement);
        }
    }

    // Find element at row r and column c
    private Element findElement(int r, int c) {
        LinkedList<Element> row = mMatrix.get(r);
        Element e = row.get(c);
        return e;
    }
}
```

(a) Give two reasons why `Element` should not have public state and provide a better mutable `Element` definition. [4 marks]

(b) Explain why `ArrayList` and `LinkedList` are appropriate choices in this context. [2 marks]

(c) Write a constructor for `SparseMatrix` that takes arguments specifying the number of rows and columns and initialises state appropriately. [2 marks]

(d) Provide the member method `get(int r, int c)`, which retrieves the value at row `r` and column `c` of the matrix, and the method `set(int r, int c, int v)`, which sets the value of it to `v`. Your methods should throw an exception if invalid arguments are supplied. [6 marks]

(e) By making `Element` objects `Comparable` show how to keep the linked lists in ascending column order and hence how to make `get()` and `set()` more efficient. If `get()` operations are more common than `set()` operations, suggest a better choice than `LinkedList` for the type of the inner list. [6 marks]