COMPUTER SCIENCE TRIPOS Part II (General)
DIPLOMA IN COMPUTER SCIENCE

Monday 5 June 1995 1.30 to 4.30

Paper 10 (Paper 1 of Diploma in Computer Science)

Answer **five** *questions.*
Submit the answers in five **separate** *bundles each with its own cover sheet.*
Write on **one** *side of the paper only.*

1 Regular Languages and Finite Automata

Describe how to derive from any regular expression a deterministic finite automaton describing the same language. [15 marks]

Justify the claim that the resulting automaton does describe the same language. [5 marks]

2 Modula-3

In the context of Modula-3, define the following: *compilation unit*, `INTERFACE`, `MODULE`, `IMPORT` and `EXPORT`. [5 marks]

The following module, which is in the file `Test.m3`, has been written to test two procedures, `Twice` and `Thrice`, which are imported through the interface `MultI` (in the file `MultI.i3`) from the module `MultM` (in the file `MultM.m3`).

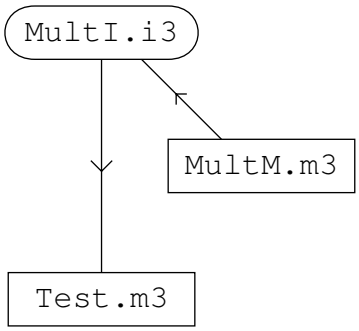
```
MODULE Test EXPORTS Main;

IMPORT MultI;

BEGIN
  MultI.Twice ('A')
  MultI.Thrice ('A')
END Test.
```

The procedures `Twice` and `Thrice` are intended simply to output the single character, presented as parameter, twice and three times respectively. Write a suitable interface `MultI` and a suitable module `MultM`. [8 marks]

Working in terms of files containing source code, the relationships between the module `Test`, the interface `MultI` and the module `MultM` may be represented by the following diagram:



In this representation, a box with rounded ends represents an interface and a rectangular box represents a module. The arrows represent the directions of imports and exports. Augment this diagram to include the file containing the interface `Main` and any other interfaces and modules used directly or indirectly by the interface and module you have written. [7 marks]

3 Further Modula-3

Synchronisation of threads in Modula-3 is achieved through the use of mutexes and condition variables. An alternative scheme would be to use Dijkstra semaphores. A semaphore has a hidden value (usually set to 1 initially) and two atomic operations:

> `wait` (sometimes called P) decrements the stored value. If the result is negative, the thread is suspended; otherwise it continues.
>
> `signal` (sometimes called V) increments the value. If there are any other threads suspended while waiting for the semaphore, one of them is allowed to continue.

Write an interface `Semaphore` defining an opaque object type `T` with `init`, `signal` and `wait` methods. [5 marks]

Sketch an implementation of the `Semaphore` module giving a concrete revelation of `T` and implementing appropriate default methods. [10 marks]

Show how the interface and implementation could be extended to derive a sub-type of `T` with an extra method, `try`, which works like `wait` but returns a `BOOLEAN` value instead of blocking. In the normal case, `try` should return `TRUE` but when the thread would have been suspended, the value in the semaphore is left unchanged and it should return `FALSE`. [5 marks]

4 Compiler Construction

Construct the characteristic finite state machine for the following grammar.

$$\begin{aligned} S &\rightarrow A\ B\ \texttt{eof} \\ A &\rightarrow A\ B \mid B\ a \\ B &\rightarrow (A) \mid b \end{aligned}$$

[6 marks]

Explain what is meant by the FOLLOW set for a non-terminal symbol in a grammar, and derive the FOLLOW sets for A and B in the above grammar. [4 marks]

Construct, with explanation, the SLR(1) *action* and *goto* matrices for the above grammar. [5 marks]

Illustrate how the SLR(1) parsing algorithm works for this grammar by showing the successive states of the parser stack and input stream while parsing

$$b\ a\ b\ (\ b\ a\ )\ \texttt{eof}$$

[5 marks]

5 Data Structures and Algorithms

For each of the following situations identify one data structure or algorithm that it would be sensible to use, and another that would in principle achieve the desired result but which would have significant disadvantages. You may identify standard methods by name and need not describe in detail how they work, but should make it clear what properties the schemes that you identify have that make some of them more appropriate than others.

(a) You need to represent some (directed) graphs where when a graph has N vertices it will have around $N \log N$ edges. The number of vertices, N, may become quite large. [4 marks]

(b) In the process of rendering a graphical image you have already sorted all the objects that have to be drawn with an ordering based on their distance from the viewpoint. Now the image has been changed slightly so that you can start to display the next frame of the video sequence, so all the distances have changed, and you need to sort the objects again. [4 marks]

(c) You need to build a table. It will be possible to insert objects into the table or retrieve previously stored ones. The only operation you are permitted to perform on objects is a pair-wise comparison that can tell if two objects are equal and if not indicates an ordering between them. There will be both plenty of insertion operations and plenty of lookups. [4 marks]

(d) You need to find the shortest distance (through a directed graph that has lengths associated with each edge) from a nominated source point A to each of a collection of destinations $\{B_i\}$. [4 marks]

(e) Blocks of material, each identified by a key, are to be stored on a large disc. From time to time new blocks (and associated keys) will need to be added, but mostly you need to service requests where a user submits a key and wants the corresponding information recovered. There is so much data that the disc is fairly full. [4 marks]

6 Data Structures and Algorithms

A reasonable approximation for the average number of probes to insert a new entry into an open hash table where a fraction β of the table is already in use is $1/(1-\beta)$.

Suppose that data is stored by initially creating a really tiny hash table (say a vector of size just 8). Entries are added to the table from time to time. Whenever the table becomes 3/4 full a new hash table, twice the size, is created: all existing data is taken out of the old table, and inserted instead into the new one. Thus in general the table that is in use will be between 3/8 and 3/4 full, and looking things up in it will be efficient.

Although the above method has good predicted costs for *retrieving* information stored in the table, there remains some worry that the repeated cost of copying data from smaller to larger tables may be excessive. Suppose that at some stage N items have been inserted and that the very last insertion provoked the copying step. Estimate the ratio between the *total* number of hash probes performed while building the table and N. How does it compare with the number that would have been used if the table had been built full-sized to start with rather than having to grow stage by stage on the way? [20 marks]

[If you really need their values, you may assume that $\ln(2) = 0.7$, $\ln(3) = 1$ and $\ln(5) = 1.6$, but note that you are not expected to perform any arithmetic tedious enough to call for a calculator: a reasonable estimate (for example to within a factor of 1.5) and a justification of how that value was arrived at is what is required.]

7 Operating System Foundations

Why are multiple buffers often used between producing and consuming processes? [4 marks]

Describe the operation of a *semaphore*. [4 marks]

What is the difference between a counting semaphore and a binary semaphore? [2 marks]

The code below is intended to synchronise buffer control between multiple producing processes and a single consuming process. `freeBuffer` and `fullBuffer` are semaphores. Describe an execution sequence which causes error.

Initialisation:

```
current_free := 0;
current_full := 0;
freeBuffer := buffers;
fullBuffer := 0;
```

Producers:

```
LOOP
     produce item;
     WAIT(freeBuffer);
     COPY (buffer[current_free], item);
     current_free := (current_free + 1) MOD buffers;
     SIGNAL(fullBuffer);
END;
```

Consumer:

```
LOOP
     WAIT(fullBuffer);
     COPY (item, buffer[current_full]);
     current_full := (current_full + 1) MOD buffers;
     SIGNAL (freeBuffer);
     consume item;
END;
```

[5 marks]

Modify the code to correct the problem. [5 marks]

8 Operating System Functions

What properties of file access and of discs does a *log-structured* file system use to optimise file system performance? Describe scenarios in which the relevant assumptions are valid and invalid. [8 marks]

Describe the operation of such a file system, including the actions of the *sweep*, or *garbage collection*, stage, and how the system recovers from crashes. [12 marks]

9 Unix Case Study

Describe the segments that compose a Unix process and how the `exec` functions create these segments from the sections of an executable binary file. [8 marks]

Using as an example the command line `ls | wc`, describe the sequence of operations and system calls that a shell must perform to execute the commands to the point at which an `exec` function is invoked on each of `ls` and `wc`. [12 marks]

10 Computation Theory

Give a brief explanation of each of the following:

(a) the encoding of register machine programs as numbers [5 marks]

(b) the notion of a *universal* register machine [3 marks]

(c) the notion of a register machine *decidable* subset of $\mathbb{N} = \{0, 1, 2, \ldots\}$ [2 marks]

Assuming the existence of a universal register machine, or otherwise, show that there is a unary computable partial function $f(x)$ such that both $\{x \in \mathbb{N} \mid f(x) \text{ is defined}\}$ and $\{y \mid \text{for some } x,\ f(x) = y\}$ are register machine undecidable. If you appeal to the undecidability of the Halting Problem, you must prove it. (You may assume the existence of register machines for coding and decoding pairs of numbers as numbers.) [10 marks]

11 Numerical Analysis I

For Single Precision in the *IEEE* binary floating point standard (*IEEE* 754) the *precision* is defined as 24, and the *exponent* requires 8 bits of storage. With reference to *IEEE* Single Precision, explain the terms *exponent*, *significand*, *precision*, *sign bit*, *normalised number*, *denormal number*, *hidden bit*. [7 marks]

How many bits are required to store a Single Precision number? How is the exponent stored? What is the value of the hidden bit for (a) normalised, and (b) denormal numbers? [3 marks]

Define the terms *absolute error*, *relative error*, *machine epsilon*. [3 marks]

Suppose $x^* = 4.0027$, $y^* = 4.0047$ are numbers represented on a computer with *machine epsilon* $= 0.5 \times 10^{-4}$. Estimate roughly (i) the maximum *absolute error*, and (ii) maximum *relative error* in evaluating $(y - x)$.

Use your results to explain the term *loss of significance*. [7 marks]

12 Digital Electronics

There exist several methods to speed up the carry completion in a digital adder. Explain one. [8 marks]

A particularly ingenious method invented in the mid-nineteenth century by Charles Babbage is called 'anticipatory carry'. Give an explanation of this method and illustrate it with an example. [12 marks]

13 Discrete Mathematics

Show that the maximum flow in a capacitated network is the capacity of the minimum cut between source and sink. [14 marks]

Exhibit the maximum matching problem in a bipartite graph as a special case of network flow. [6 marks]