3 HOURS

ANSWER ALL 4 QUESTIONS

Electronic calculators are not permitted.

Question 1

Answer any *two* parts from (a), (b), (c), (d):

Note : where an algorithm is asked for, you may write in any suitable pseudo-code.
Correct syntax for any computer language is not expected.

Part (a)

i) A **Set** may be regarded as an unordered list, in which no element may occur more than once.
Define a suitable specification for a Set, and therefore a public interface to a C++ class named Set.
[Consider only operations on a single Set, not operations combining Sets]
[2 marks]

ii) Suppose that you have decided to implement a Set using a linear array.
Using diagrams and pseudo-code, or otherwise, outline in reasonable detail an implementation of each of the requisite public functions.
[4 marks]

iii) Compare the ease of programming, and computational effort of the following operations on both a Set and a List:

- addition of an element
- removal of an element

[2 marks]

iv) How would your answers to part (iii) be modified for a linked-list implementation of the Set class ? [2 marks]

v) How would your answers to part (iii) be modified for the **MultiSet** Abstract Data Type (i.e. a Set in which multiple copies of an element are allowed)
[2 marks]

QUESTION 1 CONTINUES OVER PAGE

TURN OVER

Part b)

i) Consider a Lookup Table consisting of 5 entries stored in a **sequence**, each of which is a (key, value) pair:

(A, val_A), (B, val_B), (C, val_C), (D, val_D), (E, val_E)

If the ***unit of cost*** of retrieval is taken to be the time to move to an entry and test whether its key matches the one being searched for, determine the total cost of retrieving the values associated with the following keys, in order using **sequential search**:

CADCCBEDEEDADCC

[2 marks]

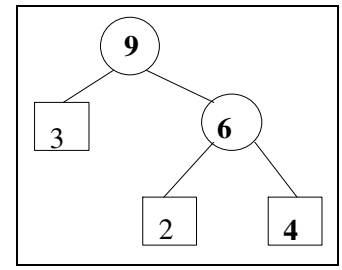
ii) Repeat part (i) using both the **move-to-front** algorithm (i.e. after each value is successfully retrieved its entry is moved to the front of the sequence) and the **migrate-to-front** algorithm (i.e. after each value is successfully retrieved its entry is swapped with the previous entry in the sequence). [7 marks]

iii) Discuss how each of the two algorithms in part (ii) would be implemented if the sequence were based on a) an array, b) a linked-list. In each case comment on the efficiency of the implementation. [3 marks]

QUESTION 1 CONTINUES OVER PAGE

c)

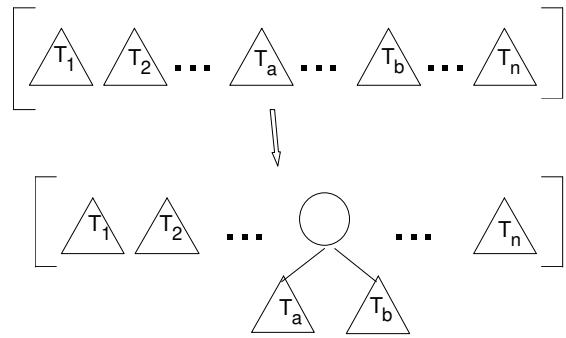
A *weighted binary tree* is defined as a binary tree wherein each node has a weight, and the weight of a parent node is the sum of the weights of its two children. An example with five nodes is shown on the right. The *root-weight* of such a tree is the weight associated with the root node (9 in the case of this example).



Huffman's algorithm is defined as follows:

Given a forest of weighted trees $\{T_1, T_2, \ldots T_n\}$, remove the two trees T_a and T_b with the lowest weight of their root nodes and replace with a weighted tree having T_a, and T_b as the two children of its root.

This is illustrated on the right.



+

It is desired to find simple codes of 1's and 0's for the letters A to E so that a string of these letters can be unambiguously coded with the shortest length.

For the particular string

CADCCBEDEEDADCC

carry out the following steps :

i) Construct a forest of 5 single-node trees with root weight representing the frequency of each letter in the string. [2 marks]

ii) Apply Huffman's algorithm repeatedly until only one tree remains in the forest. [6 marks]

iii) By assigning 0 to a left branch and 1 to a right branch derive a code for each of the letters in the string. Thus give the code for the complete string. [2 marks]

iv) Given the codes for each letter, and the weighted tree from which they are derived, describe how to decode the stream of 1's and 0's to give back the original string of letters. [2 marks]

QUESTION 1 CONTINUES OVER PAGE

d)

i) What properties should a good hashing function possess ?

The following C++ code is proposed for hashing character strings, that are known to consist exclusively of lower-case letters, into a table of size M:

```
int intval(char c) {          // intval maps 'a' to 0,
                              // 'b' to 1, ... 'z' to 25
        return c - 'a';
}
int h (char *s, int M) {
        int sn = strlen(s);
            return (intval(s[0]) + sn)%M;
}
```

How well does this function satisfy your criteria ? [3 marks]

ii) suppose M = 17 and the table entries are stored in positions 0 to 16 of an array. Draw diagrams to show the state of the table after the words of the following sentence are added, in the order that they occur:-

"life is a tale told by an idiot full of sound and fury signifying nothing"

assuming that hashing conflicts are removed by a) chaining, b) open addressing [7 marks]

iii) In general, why might open addressing be preferred to chaining as a conflict resolution strategy ? What are its disadvantages ? [3 marks]

END OF QUESTION 1

Question 2

Answer any *two* parts from (a), (b) and (c).

a) This part is concerned with yes-no problems. Let A, B and C denote arbitrary yes-no problems.

i) Define what it means to say that A *reduces to B in p-time* (in symbols, $A \leq B$). [4 marks]

ii) Let $\leq$ be the relation defined in part (a) (i). Prove that $\leq$ is *transitive* i.e. if $A \leq B$ and $B \leq C$ then $A \leq C$. [4 marks]

iii) Let NP denote the class of yes-no type problems that can be solved by a non-deterministic Turing machine in polynomial time. Using the ordering $\leq$ defined in part (a) (i), or otherwise, define the class of all *NP complete problems.* [4 marks]

b)

i) Explain what it means when we say that a yes-no problem A is *undecidable.* [4 marks]

ii) Explain clearly what the halting problem is. [4 marks]

iii) Sketch in outline a proof that the halting problem is undecidable. [4 marks]

c)

i) Define the *Travelling Salesman Problem* (TSP) and the *Hamiltonian Circuit Problem* (HCP). [4 marks]

ii) Assume that the HCP is NP-complete (i.e. you do *not* have to prove this). Use that assumption and the transitivity of $\leq$ to prove that TSP is also NP-complete. [8 marks]

END OF QUESTION 2

Question 3

D0B 1996

Answer All parts a, b and c.

a) Explain the terms *lexeme and token,* giving examples. [3 marks]

b) Explain what is meant by a *recursive-descent* compiler. [3 marks]

c) Suppose it is desired to add a new construct, the **until-do,** to the C/C++ programming languages. For example in

```
until (x>1024) do
        x = x * 2;
print (x);
```

if **x** is not greater than 1024 then the assignment statement x = x * 2; is repeatedly executed until the expression evaluates to true, otherwise it is not executed at all. The value of x is then printed. (For the sake of simplicity in this question it is assumed that a primitive print statement has also been added to the language.)

Your task in the following parts of the question is to generalise from and develop this informal example into a clear specification and implementation through the use of syntax diagrams.

It is part of *your* task to clarify any vagueness in this question in a way which is consistent with the C/C++ languages.

i) Modify the code given above so that it would print the value of x on each iteration.

ii) Show the kind of code which might be generated by a simple compiler for a stack machine from your modified code.

iii) Draw and annotate a syntax diagram for the **until-do** statement. You may assume that syntax diagrams for **expression** and the **assignment** and **print** statements exist.
State clearly any further assumptions you need to make, taking note of the task described to you in the preamble to this part of the question.

iv) Sketch an implementation of your syntax diagram as a procedure (function) in a recursive-descent compiler. Briefly describe the purpose of any other procedures or variables to which your implementation refers. In the case of variables, for each one you should make it quite clear whether it is local or global, and explain why. You may refer to **newlabel(),** a function which returns a unique label each time it is called.
Your implementation should work with nested constructs, and you should justify how your design allows this. [19 marks]

END OF QUESTION 3

CONTINUED

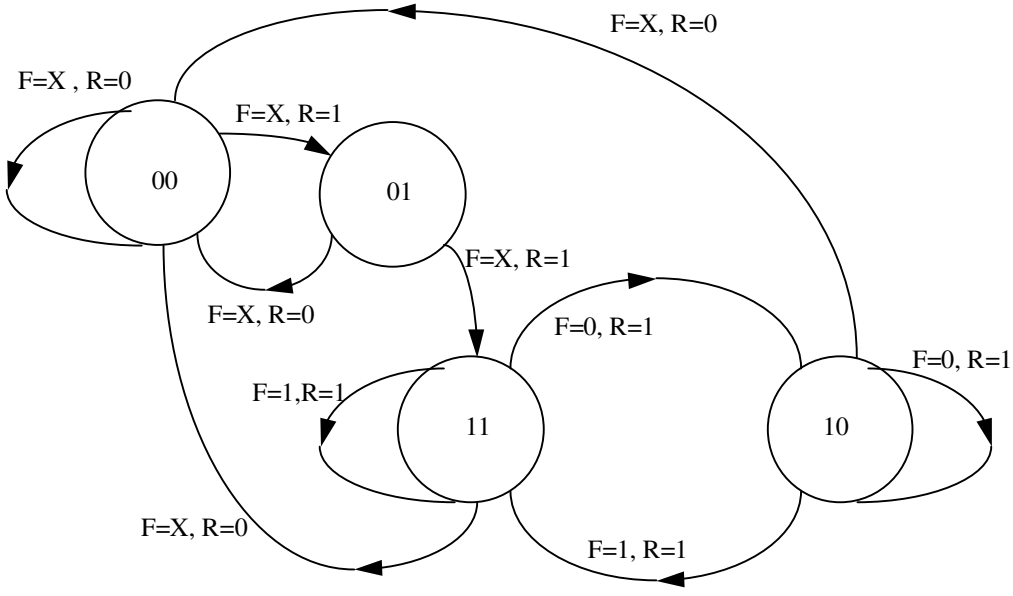
Question 4

D0B 1996

Answer 3 parts from parts (a)-(f):

(a) Outline the structure and operation of Finite State Machines as used in digital electronic circuits and explain why they are useful.

(b) Draw the truth table for the Finite State Machine (FSM) with the following State Diagram:



Use Karnaugh maps or other technique to derive the equations for the next state bits.

Draw the final circuit for the Finite State Machine using AND, OR and INVERTER gates, and flip-flops for the state store.

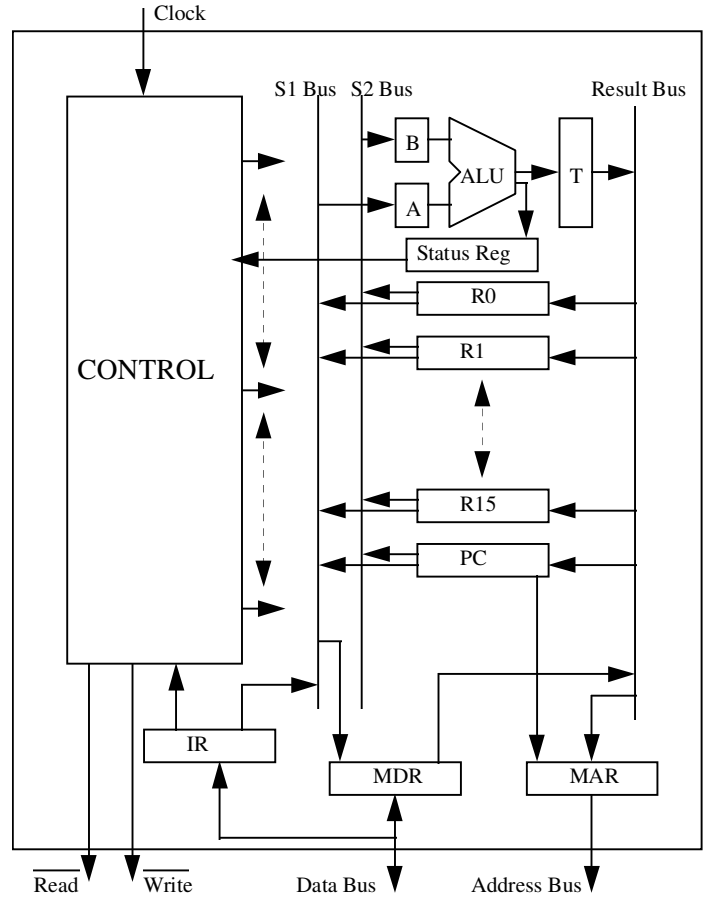
(c) Show how a 2x2 bit memory can be built from one-bit transparent flip-flops and tri-state devices, and briefly discuss its operation.

(d) Draw a diagram of the physical structure of either a p-n-p or a n-p-n transistor, and describe its operation.

QUESTION 4 CONTINUES OVER PAGE

(e) What is the role of transistors in digital electronic circuits, in particular CMOS circuits. Illustrate your answer with a description of a CMOS NAND or NOR Circuit.

(f) Describe the function of the major elements of the CPU in the diagram below and give a general outline of its operation.



END OF QUESTION 4

END OF PAPER