Section III

20 marks
Attempt either Question 24 or Question 25
Allow about 35 minutes for this section

Answer the question in a SEPARATE writing booklet. Extra writing booklets are available.

If you include diagrams in your answer, ensure that they are clearly labelled.

Marks

Question 24 — Evolution of Programming Languages (20 marks)

(a) A need for greater productivity has influenced the development of different paradigms.

(i) Discuss the main influences on a programmer's productivity. **4**

(ii) Discuss other factors that have influenced the development of paradigms. **4**

(b) Inheritance, encapsulation and polymorphism are all aspects of object-oriented programming languages. Define each of these aspects and explain how they improve the reusability and maintainability of code. **6**

(c) A programmer has been asked to develop a system for a doctor to help in the selection of antibiotic medicines for hospital patients who have certain bacterial infections. The system uses data about the bacteria, patient history, symptoms and laboratory test results to suggest the type and dosage of antibiotic. **6**

Recommend the most appropriate paradigm for the solution of this problem. Explain why the other paradigms would be less appropriate.

OR

24.

a) (i) The main influences ~~of~~ on programmer's productivity include:

(i) Speed of code generation – the faster a programmer is able to generate code, the more productive the paradigm is. For example, coding in the imperative paradigm is much slower than coding in the functional (which has a much simpler syntax.

(ii) Approach to testing:– the more easily a program's errors can be isolated, the more productive the programmer will be (as the errors can be detected & removed efficiently). For example, it is easier to debug OOP than imperative languages as OOP is founded on encapsulated objects

(iii) Effective maintenance – similarly, the easier it is to maintain and update a product, the ~~easier it is to use~~ more productive the programmer can be. For example, it is much easier to maintain logic programs over any other paradigm as new facts & rules can be simply appended to the existing ones.

(iv) ~~Sp~~ Efficiency of solution once coded – ~~the~~ the more efficient the compiled code is, the more productive the programmer. eg. imperative is much more efficient than logic ~~as~~ (on a single processor machine) as the programmer specifies the exact input, processing & output rather

than relying on heuristics or permutations.
(v) Learning curve — the easier a programming paradigm is to understand (eg. logic) the more productive the programmer will be.

(ii) Other factors:
(i) Recognition of repetitive tasks to form routines — this greatly provided the impetus for developing the Object Oriented paradigm as it encourages modularity and the robustness of code.
(ii) A desire to solve different problems — the desire to program AI has accelerated the development of functional & logical paradigms; and it has also shown a shift away from the Von Neumann model in an attempt to create A I.
(iii) A recognition of different building blocks — such as facts and rules, mathematical functions, objects & classes — have ~~created~~ influenced the development of logic, functional & OOP paradigms respectively.
(iv) Emerging technologies such as ~~[illegible]~~ parallel computing has influenced the development of ~~[illegible]~~ functional / logic paradigms as they

can solve problems much faster by ~~[illegible]~~ sharing the processing through multiple computers. The development of visual programming languages has also shown that the object oriented program is very useful in creating reusable visual controls.

(b) Inheritence – the ability of objects to take on the characteristics (attributes & methods) of their parent class or classes. It improves the reusability & maintainability of code by making the code robust and very modular – there is very little redundant data as similar objects have been predifined through classes. Thus to ~~[illegible]~~ update a class of objects, only that class needs to be modified, not the entire selection of objects individually.
Encapsulation – the process of hiding an objects attributes and methods from the environment. This means that only the object's own methods can alter its own data. This greatly assists reusability & maintainability as the ~~[illegible]~~ objects are

self contained objects of code. They are defined as abstract data types and encapsulation ensures that they must be able to function on their own, greatly increasing its potential for reusability & ~~upgrading~~. upgrading.

Polymorphism - the ability to appear in different forms - ie. ~~different~~ runtime methods will process data differently depending on the number & type of parameters passed to it. This increases reliability as it allows ~~the~~ one method to cater for various conditions of the passing parameters (making the module more universal). It also assists understandability as different data processing can be added ~~[illegible]~~ to the same method by simply varying parameters.

c) The logic paradigm would be the most appropriate paradigm because:

- an expert system could be developed
- the facts and rules regarding the patients and the varying diseases could be stored in a knowledge base.
- the inference engine can ~~[illegible]~~ apply the knowledge ~~[illegible]~~ the knowledge base to resolve the goals of ~~selecting~~ ~~[illegible]~~ determining whether a particular antibiotic is appropriate
- ~~[illegible]~~ the expert system could be used to develop heuristics and thus ~~generate~~ generate the best possible recommendation by probabilities ("rules of thumb") ~~[illegible]~~ thus select the most appropriate antibiotic for the symptoms of the virus

Thus logic seems to be the most appropriate paradigm.

Other paradigms —

Imperative — inappropriate as it would require too much time to actually develop the code ~~to eat~~ as the programmer must specify the sequence of processes to diagnose & select the antibody.

OOP — inappropriate as it too is a sequential paradigm. Objects are too vague as their are only ~~two~~ 3 possible objects : patients, bacteria, antibiotics. This is too general & it would ~~&~~ be hard to develop a method for selecting the most appropriate antibodies.

Functional — the system isn't based on mapping domain against range as their are so many variables such as the varying symptoms & the varying antibodies — ~~a mate~~ a mathematical relationship may be very hard to develop.

∴ Logic is most appropriate.