Q1 a

Q2 a

Q 3 b

Q4 c

Q 5 c

Q 6 a

Q 7 b

Q 8 b

Q 9 b

Q 10 a

11.What are the symptoms of the present software crisis? What factors have contributed to the making of the present software crisis? What are possible solutions to the present software crisis? (2 M)

Ans: The symptoms of the present software crisis are increasing cost of software products, unreliable products that are full of bugs, and products delivered late. (1 M)

The factors contributing to software crisis are the increasing complexity of software products being developed, lack of professionals adequately trained in software engineering techniques, and lack of progress of the software engineering discipline itself. (0.5 M)

The possible solutions to the presenty software crisis are adequate training in software engineering techniques and progress of the software engineering discipline itself. (0.5 M)

12.Explain why the effort, time, and cost required to develop a program using the build and fix style increase exponentially with the size of the program? How do software engineering principles help tackle this rapid rise in development time and cost? (2 M)

Ans:The effort, time, and cost required to develop a program using the build and fix style increase exponentially with the size of the program primarily due to the human cognitive limitations. (1 M)

Software engineering principles overcome this problem through abundant use of the abstraction and decomposition. (1 M)

Full marks will be given only if the correct answer is given. Else, if the related answer is given, only half the marks will be awarded.

13. What are the three basic types of program constructs necessary to develop the structured program for any given problem? Give examples of these constructs in 'C' language.

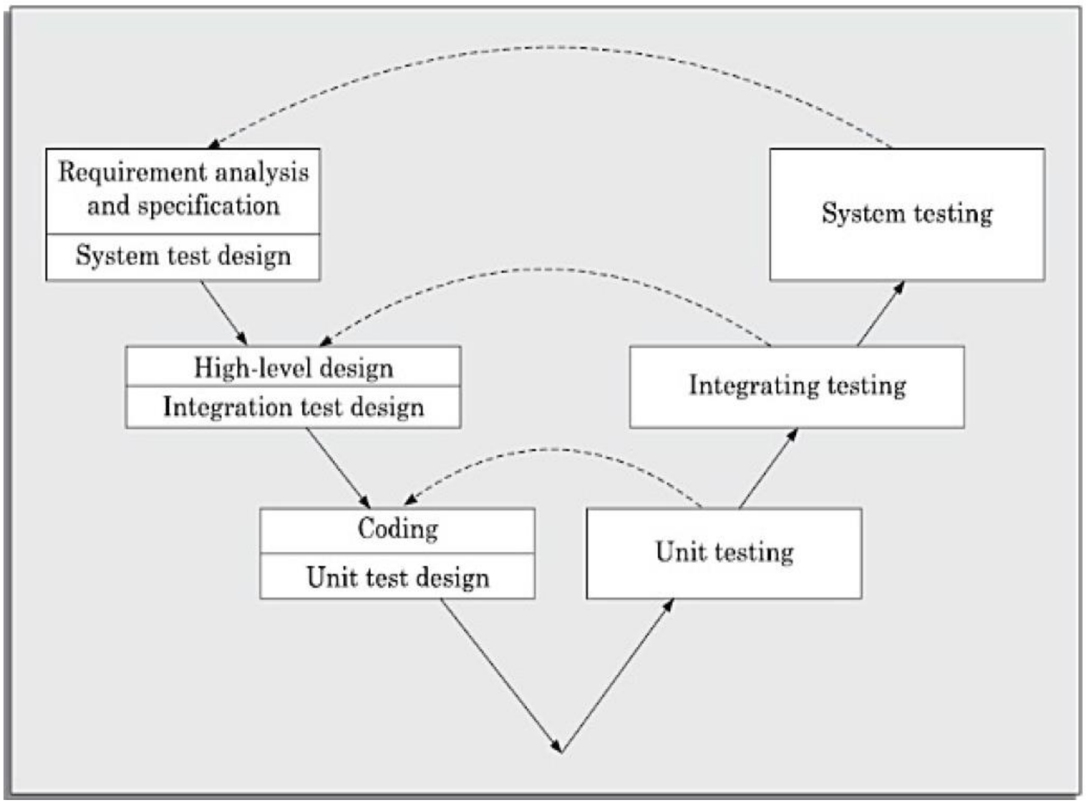
Ans: The three basic types of program constructs necessary to develop a sturcutured program are:

- Sequence
- Selection
- Iteration (1 M)

Examples of these constructs from "C" language:

- Sequence: a=b;
- Selection: if(a≤b) b=0 else c=0;
- Iteration: while(a≤b) a++; (1 M)

14 With the help of a diagram describe the phases involved in the V-Model.Justify how this model is different from the Waterfall Model 3M



1M

As shown in Figure 2.5, there are two main phases—development and validation phases. The left half of the model comprises the development phases and the right half comprises the validation phases.

- In each development phase, along with the development of a work product, test case design and the plan for testing the work product are carried out, whereas the actual testing is carried out in the validation phase. This validation plan created during the development phases is carried out in the corresponding validation phase which have been shown by dotted arcs in Figure 2.5.
- In the validation phase, testing is carried out in three steps—unit, integration, and system testing. The purpose of these three different steps of testing during the validation phase is to detect defects that arise in the corresponding phases of software development—

  requirements analysis and specification, design, and coding respectively.

(Each point carries 1/2M. ½ X 2 =1M)

V-model *versus* waterfall model

We have already pointed out that the V-model can be considered to be an extension of the waterfall model. However, there are major differences between the two. As already mentioned, in contrast to the iterative waterfall model where testing activities are confined to the testing phase only, in the V-model testing activities are spread over the entire life cycle. As shown in Figure 2.5, during the requirements specification phase, the system test suite design activity takes place. During the design phase, the integration test cases are designed. During coding, the unit test cases are designed. Thus, we can say that in this model, development and validation activities proceed hand in hand.

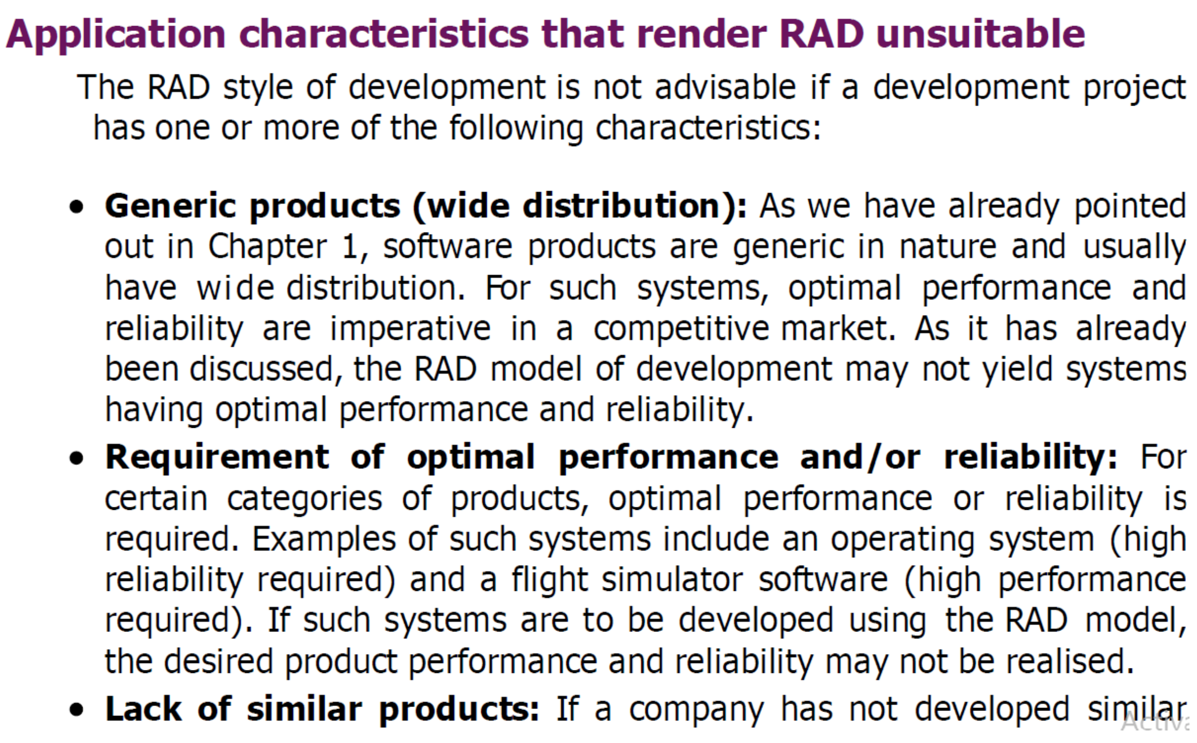
(1M)

Q15 Elucidate the characteristics of the RAD model that make it suitable and unsuitable for certain applications of Software Development. Substantiate your answer using 3 points each. 3M

Applicability of RAD Model

The following are some of the characteristics of an application that indicate its suitability to RAD style of development:

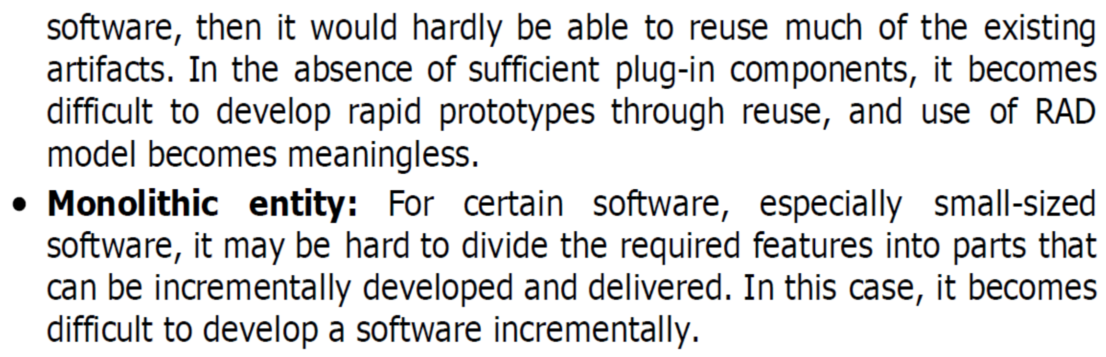
- **Customised software:** As already pointed out a customised software is developed for one or two customers only by adapting an existing software. In customised software development projects, substantial reuse is usually made of code from pre-existing software.
- **Non-critical software:** The RAD model suggests that a quick and dirty software should first be developed and later this should be refined into the final software for delivery. Therefore, the developed product is usually far from being optimal in performance and reliability. In this regard, for well understood development projects and where the scope of reuse is rather restricted, the Iiterative waterfall model may provide a better solution.
- **Highly constrained pro ject schedule:** RAD aims to reduce development time at the expense of good documentation, performance, and reliability. Naturally, for projects with very aggressive time schedules, RAD model should be preferred.
- **Large software:** Only for software supporting many features (large software) can incremental development and delivery be meaningfully carried out.



Application characteristics that render RAD unsuitable

The RAD style of development is not advisable if a development project has one or more of the following characteristics:

- **Generic products (wide distribution):** As we have already pointed out in Chapter 1, software products are generic in nature and usually have wide distribution. For such systems, optimal performance and reliability are imperative in a competitive market. As it has already been discussed, the RAD model of development may not yield systems having optimal performance and reliability.
- **Requirement of optimal performance and/or reliability:** For certain categories of products, optimal performance or reliability is required. Examples of such systems include an operating system (high reliability required) and a flight simulator software (high performance required). If such systems are to be developed using the RAD model, the desired product performance and reliability may not be realised.
- **Lack of similar products:** If a company has not developed similar



software, then it would hardly be able to reuse much of the existing artifacts. In the absence of sufficient plug-in components, it becomes difficult to develop rapid prototypes through reuse, and use of RAD model becomes meaningless.
- **Monolithic entity:** For certain software, especially small-sized software, it may be hard to divide the required features into parts that can be incrementally developed and delivered. In this case, it becomes difficult to develop a software incrementally.

(student may write any 3 points for suitability and unsuitability)

Q16 Compare the characteristics of understandability and maintainability in a good software design. How do they contribute to the long-term success of a software project? -3M

Solution:

- **Understandability** refers to how easily the design can be understood by developers and stakeholders. A design that is easy to understand facilitates smooth communication among team members and allows for accurate implementation, which in turn reduces errors.
- **Maintainability** focuses on how easily changes can be made to the design after its initial implementation. As software evolves, new requirements or bug fixes arise, making it crucial for the design to support modifications without extensive rework.

Both characteristics are essential for the long-term success of a software project. Understandability ensures that new team members or stakeholders can quickly grasp the design, leading to efficient onboarding and collaboration. Maintainability, on the other hand, enables the software to adapt to future requirements, which is critical for its longevity and continuous improvement. Together, they contribute to reduced development time, fewer errors, and easier updates.

Q17What is *logical cohesion and temporal cohesion*? Provide an example where these can be applied in software design? -3M

Solution:

Logical cohesion occurs when a module contains functions that perform similar types of operations, such as data input, error handling, or report generation, but the exact task to be performed is selected by the user or determined by the program at runtime.

Example: A module containing different print functions for generating various reports, such as salary slips, grade sheets, and annual reports. While these functions are logically related as they all perform printing tasks, they handle different types of data and purposes, leading to logical cohesion.

Temporal cohesion occurs when the functions of a module are grouped together because they are executed at the same time, such as during system initialization, setup, or shutdown. These functions are related only by their timing of execution, not by their purpose.

Example: A module that handles booting up a computer might include functions such as memory initialization, device setup, and loading the operating system. Though these functions perform different tasks, they are all required to be executed during the startup process, leading to temporal cohesion.

Q 18 Three Functional requirements (3 M)

Q 19 Decision Tree (Correct format 2M)

Decision Table(Correct format 2 M)