**Question**

1(a) i Describe by using a simple diagram the concept of formal methods

1(a) ii What are the major reasons for considering formal methods.

1(b) Differentiate between functional and non-functional requirement in formal methods.

2(a) List and discuss briefly the recommended development process of software engineering methodology.

2(b) Using spectrum of rigor, discuss what is meant by formal methods.

3(a) i Differentiate between propositional logic and first order predicate logic.3(a) ii Using propositions A, B, and C; discuss any three basic logical operations in propositional logic

3(b) Represent the text “***Every man is mortal. Smith is a man***. ***Therefore, Smith is mortal****”* in first order predicate calculus expression.

4(a) Using the principles of first order predicate calculus to represent this statement

***“There exists an object that is either a curious monkey or not a monkey at all”***.

4(b) List and discuss any three (3) types of Formal Specification

5(a) Define the following:

(i) A well-formed formula (ii) A quantifier (iii) A Predicate (iv) A Term

5(b) Translate the following Predicate Calculus to statements

**(i) GREATER (a, b) = T, if a < b**

**= F, otherwise.**

**(ii) (ꓯ y) LIKE (Mother (y), y).**

6(a) Discuss with examples any three (3) software development strategies

6(b) What do you understand by requirement document

7(a) Discuss the structure of a requirement document

7(b) Enumerate the steps involved in writing a system requirement specification

ANSWER

1(a)i Formal method refers to a method which analyzes and studies the formal structure of thinking in the science of logic. It is a scientific methodology of modern logic system, representing the creativity of modern logic thoughts. Besides, it is also a framework tool for constructing the modern logic theory, and an intermediary linking theoretical logic and specific logic. For the formal method, thinking forms are compared with different contents (mainly the proposition and reasoning) to find the connection methods among all parts.

1(a)ii The major reasons for considering formal methods are:

* Systems are becoming increasingly dependent on software components;
* Complexity of Systems with embedded software has increased - A modern mid-sized sedan car is equipped with at least 150 processors;
* Maintaining reliability in software-intensive systems is very difficult. Quality problems with software may cause minor irritations or major damage to a customer’s business including loss of life.

1(b) Key differences are:

* A functional requirement defines a system or its component whereas a non-functional requirement defines the performance attribute of a software system.
* Functional requirements along with requirement analysis help identify missing requirements while the advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software.
* Functional Requirement is a verb while Non-Functional Requirement is an attribute
* Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc. whereas transaction corrections, adjustments, and cancellations, Business Rules, Certification Requirements, Reporting Requirements, Administrative functions, Authorization levels, Audit Tracking, External Interfaces, Historical Data management, Legal or Regulatory Requirements are various types of functional requirements.

2(a) The recommended development process of software engineering methodology are:

* Waterfall: When it comes to software development, Waterfall is the most traditional and sequential choice. Waterfall requires plenty of structure and documentation up front. It is divided into self-contained stages or steps. The first stage is vital, requiring a full understanding by both developers and customers of the project’s demands and scope before anything begins. The stages are relatively rigid and often follow this sequence: determine the project’s requirements and scope, analyze those requirements, design, implement, test, deploy and finally, maintain.
* Agile: The Agile methodology was developed as a response to growing frustrations with Waterfall and other highly structured, inflexible methodologies. This approach is designed to accommodate change and the need to produce software faster. Agile values individuals and their relationships and interactions over tools; it features customer collaboration throughout the development process; it responds to change instead of following a set-in-stone plan; and it focuses on presenting working software, rather than documentation.
* Feature-Driven Development(FDD): An iterative and incremental approach to software development. (FDD) is derived from the Agile methodology and is considered one way to implement it. FDD still focuses on the goal of delivering working software frequently and is an especially client-centric approach, making it a good fit for smaller development teams. To produce tangible software often and efficiently, FDD has five steps, the first of which is to develop an overall model. Next, build a feature list and then plan by each feature. The final two steps—design by feature and build by feature—will take up the majority of the effort. At each step, status reporting is encouraged and helps to track progress, results, and possible errors.
* Scrum: Another way to implement the Agile approach, Scrum borrows from Agile’s foundational beliefs and philosophy that teams and developers should collaborate heavily and daily. With Scrum, software is developed using an iterative approach in which the team is front and center—experienced and disciplined workers on smaller teams might find the most success with this method, as it requires self-organization and self-management.
* Extreme Programming: Another Agile framework, Extreme Programming (or XP) focuses on producing higher quality software using the best practices in software development. As with most Agile approaches, XP allows for frequent releases in short development sprints that encourage change when needed. In general, XP follows a set of values, rather than steps, including simplicity (develop what is required, nothing more); communication (teams must collaborate and work together on every piece of the software); consistent feedback; and respect.
* Lean: Lean is at once a workflow methodology and a mindset, incorporating principles and practices from the manufacturing space and applying them broadly to a variety of industries, including software development. While Agile is an excellent methodology for the practical application of development best practices, it does not include instructions for scaling these practices across the organization or applying them outside of development-type work. Lean’s basic principles—optimize the whole, eliminate waste, build quality in, create knowledge, defer commitment, deliver fast, and respect people—can help to guide decision-making across the organization in a way that can help to unearth potential issues and maintain a healthy organizational culture.

2(b)

3(a)i **Key differences between PL and FOL are:**

* Propositional Logic converts a complete sentence into a symbol and makes it logical whereas in First-Order Logic relation of a particular sentence will be made that involves relations, constants, functions, and constants.
* The limitation of PL is that it does not represent any individual entities whereas FOL can easily represent the individual establishment that means if you are writing a single sentence then it can be easily represented in FOL.
* PL does not signify or express the generalization, specialization or pattern for example ‘QUANTIFIERS’ cannot be used in PL but in FOL users can easily use quantifiers as it does express the generalization, specialization, and pattern.

3(a)ii