Q1

SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use. SDLC uses different models to achieve their desired output. Traditional programming is when a programmer manually provides specific instructions to the computer based on their understanding and analysis of the problem. If the data or the problem changes, the programmer needs to manually update the code and execute the changes required.

Q2

* Phase 1: Planning Phase- identify the problem and scope of any existing systems, as well as determine the objectives for their new systems.
* Phase 2: Feasibility - includes gathering all the specific details required for a new system
* Phase 3: Design and Prototyping -designing of the user interface and networking to see what the system will look like
* Phase 4: Software Development - front and back-end development of a working system
* Phase 5: Software Testing Phase - ensures the application’s features work correctly and coherently and fulfil user objectives and expectations.
* Phase 6: Implementation and Integration - it’s time to make it available to its end users and deploy it to the production environment
* Phase 7: Operations and Maintenance Phase - maintenance stage, where the software is already being used by end-users.

Q3

Agile methodologies are best suited for projects that require flexibility and adaptability, allows quick response to changing requirements and to incorporate new ideas and feedback as they arise while waterfall methodologies are best suited for projects that require a detailed plan and a structured approach with extensive documentation required.

Waterfall is best for projects with concrete timelines and well-defined deliverables. If your major project constraints are well understood and documented, Waterfall is likely the best approach. The Agile methodology was created for projects where the significant constraints are not well understood.

Q4

Requirements Engineering is when a problem is well defined, and the solution is both accurate and effective. It transforms a real-world problem into a highly functional software solution.  It is an important phase in the software development life cycle because it helps to ensure that the final product meets the needs of the stakeholders and users. This process involves communicating effectively, listening patiently and documenting well and a repeat till the desired outcome has been reached.

Q5

Modularity is breaking down complex systems into smaller, manageable modules or components. These modules function independently while also collaborating seamlessly within the larger software system. This approach simplifies the addition of new features, enhancements, and the scaling of the software.

Q6

Testing is crucial as it helps in reducing project risk, providing confidence in the software quality, meeting compliance needs, ensuring satisfied users, enabling continuous improvement, and reducing overall costs.

Unit tests are very low level and close to the source of an application. They consist in testing individual methods and functions of the classes, components, or modules used by your software.

Integration tests verify that different modules or services used by your application work well together

Functional tests focus on the business requirements of an application.

End-to-end testing replicates a user behaviour with the software in a complete application environment. It verifies that various user flows work as expected

Acceptance testing, which is used to evaluate whether a specification or the requirements are met as per its delivery.

Q7

Version control is a method of tracking changes to documents and files to always know which version is the current iteration. It also enables you to maintain old versions in case you want to see what's changed or need to restore a previous version.

Q8

Resource capacity – if there are not enough resources it will cause delays in project deployment and budget constraints. End user training as well as risk management.

Q9

To keep the solution in a proper state of functioning and to avoid critical implications that will hinder it being effective. This includes improving the software overall, correcting issues or bugs

Q10

Software engineering presents unique ethical challenges due to its complexity and the difficulty in spotting issues in advance. A software engineering code of ethics is a moral compass, guiding professionals to make responsible decisions and ensuring that the products and services they develop align with societal values and expectations. At each stage, ethical considerations should be considered to ensure that the software created is reliable secure and respects user privacy. Software engineers must ensure their software is reliable, safe, and meets users’ needs.

They must also be accountable for the consequences of their actions and decisions. This includes addressing and rectifying any issues or vulnerabilities that may arise in the software.

Software engineers should strive to create inclusive and accessible software that does not discriminate against individuals based on race, gender, age, or disability.

They should also ensure that the software they develop adheres to legal and ethical standards, promoting fairness and justice in their work.

 Sources:

<https://www.institutedata.com/blog/software-engineering-code-of-ethics/>

Professional and Ethical dilemmas in software engineering by Brian Berenbach, *Siemens Corporate Research*