1. What is Software? Types of Software with Example

Software is more than just a computer program.
It includes:

- **Programs** (the code that runs),
- **Procedures** (how to use it),
- **Documentation and Data** (manuals, help files, settings, etc.)

It helps the computer system to work and perform tasks.

Types of Software

There are **two major types** of software:

1. Generic Software (Buy)

- This type is **already developed** and sold to many people.
- It is called **Commercial Off-The-Shelf software (COTS)**.
- Same software is used by thousands of users.
- It may not work exactly how you want it to.
- **Example:** Microsoft Word, Adobe Photoshop

2. Customized Software (Build)

- This is made **specifically for one user or organization**.
- It meets special needs of that customer.
- It is also called **Bespoke software**.
- **Example:** A bank may get special software built just for its internal operations.

Buy & Build Together

- Most of the time, personal users use **COTS software**.
- But companies may **buy and then customize** it for their needs.

2. What is Software Quality?

- According to standards:
  - **ISO/IEC 9126:** Software quality means how well a product meets user needs.
  - **IEEE Std 610:** It's about meeting both requirements and user expectations.

1. Functionality

Does the software do what it's supposed to do?

- Measures **how well** the software performs its required tasks.
- Includes things like correctness, security, and interoperability.

- **Example:** A banking app should correctly transfer money between accounts.

2. Reliability

Can the software be trusted to work under different conditions?

- Tells us how **dependable** the system is.
- How often does it fail? Can it recover from errors?
- **Example:** Autopilot software must not crash during flight, even if there's a minor fault.

3. Usability

Is the software easy to use and understand?

- Focuses on **user experience** – ease of learning, using, and navigating the system.
- **Example:** A mobile app with a clean interface and intuitive icons is more usable.

4. Efficiency

Does the software use system resources wisely?

- Looks at **performance** – like response time, memory usage, etc.
- **Example:** A photo editing app that runs smoothly without slowing down your device is efficient.

5. Maintainability

Can the software be easily fixed, updated, or improved?

- Concerns how **easy it is to modify** the system after it's deployed.
- **Example:** If you can easily fix a bug or add a feature without breaking other parts, that's maintainability.

6. Portability

Can the software work on different devices or environments?

- Measures how easily it can be transferred from one hardware or software environment to another.
- **Example:** A website that works on Chrome, Firefox, mobile, and desktop has high portability.

Q2. What are the challenges in software projects? Explain.

Answer:

In software projects, the main goal is to deliver the product **on time**, **within budget**, and with the **expected quality**. But achieving this is not easy. There are several challenges that make software development difficult.
The main challenges are:

1. Time

- Projects are often **delayed** and not completed on time.
- Managing schedules and meeting deadlines is a big issue.

2. Cost

- Many software projects become **too expensive**.
- They go **over budget** due to poor planning or unexpected problems.

3. Scope (Quality)

- Sometimes, the **final product does not meet quality expectations**.
- It may have **many bugs or missing features**.
- This happens due to unclear requirements or lack of proper testing.

Summary
These three major issues—**time**, **cost**, and **quality**—are known as the **Project**

Management Triangle.
Balancing all three is the biggest challenge in software developme

Give examples of software defects.

Answer:

A software defect is a problem or error in the software that causes it to behave unexpectedly or fail. Defects can be **very dangerous and costly**, especially in critical systems.

Here are two real-life examples of software defects:

1. Therac-25 Radiation Machine (1986)

- It was a **radiation therapy machine** used to treat cancer.
- Due to a **software defect**, the machine gave patients **massive overdoses of radiation**.
- This caused **serious injuries and deaths**.
- The bug was due to poor testing and no proper safety checks in the software.

2. Ariane 5 Rocket Failure (1996)

- The **Ariane 5 rocket** exploded just **40 seconds after launch**.

- Cause: A **software bug** in the navigation system.
- The system tried to convert a large number into a small memory space, which caused a crash.
- The failure cost **over $370 million**.

Conclusion:

These examples show that **software defects can lead to loss of money, property, and even human lives**. So, quality assurance and testing are very important in software development.

Q4. What is Software Testing? What are its goals and levels?

Answer:

What is Software Testing?

Software testing is the process of **executing a program** to find **errors or bugs**.
It checks if the software works correctly and meets user requirements.

Goals of Software Testing:

1. **To check if the software meets its requirements.**
   – Make sure the software does what it's supposed to do.
2. **To find errors and bugs.**
   – Testing helps locate problems so they can be fixed early.
3. **To test performance in real situations.**
   – For example, testing how an autopilot system works under stress or emergency.

Levels of Software Testing:

There are **3 main levels** of testing:

1. Unit Testing

- Tests **individual parts** or modules of the software.
- Example: Testing a single function like “login”.

2. Integration Testing

- Checks if **different modules work together** properly.
- Example: Does the login screen connect correctly to the user database?

3. System Testing

- Tests the **complete software as a whole**.
- It checks the full system behavior and performance.

Conclusion:

Software testing is a key part of software quality. It helps deliver **reliable and correct software** to users by finding and fixing problems early.

Q5. What is the role of testing in software development?

Answer:
Testing plays a very important role in software development. It helps ensure that the software is **correct, reliable, and ready for use**.

There are **two main types of testing activities** based on how the testing is done:

1. Static Analysis (Without Running the Program)

- In this method, we **check the code manually** or using tools without running it.
- This includes:
    - **Code reviews**
    - **Walkthroughs**
    - **Formal inspections**
- It is **manual and time-consuming**, but good for finding logical or design errors early.

2. Dynamic Analysis (Running the Program)

- In this method, the program is **executed with input values** to check its behavior.
- It helps find **runtime errors** like crashes, wrong output, etc.
- It is usually **automated and faster**, but may not catch all errors.

Best Practice: Use Both

- To get the best results, **both static and dynamic analysis** should be used.
- This way, we can **catch more defects** and improve software quality.

Conclusion:

Testing helps to find bugs early, reduce risks, and improve the **overall quality of the software**. It is a **key activity** in every phase of software development.

Q6. What is Software Quality Assurance (SQA)?

Answer:
Software Quality Assurance (SQA) is a set of **planned and systematic activities** that make sure the software **meets quality standards and works properly**.

It is not just testing — SQA covers the **entire software development process** to make sure the product is being built correctly at every stage.

Main Points about SQA:

1. **Planned & Structured**
    - SQA is done using a proper plan and follows rules, methods, and standards.
2. **Monitoring the Process**
    - It checks whether the team is following the correct steps while building the software.
3. **Like an Umbrella**
    - SQA covers **all phases**: requirement, design, coding, testing, and release.
    - It includes everything from **process checks to reviews and audits**.
4. **Goal**
    - To make sure that the final software product is **high-quality**, **meets user needs**, and is **free of defects**.

Conclusion:

SQA helps in building **reliable and quality software** by focusing on **process improvement** and **early detection of issues** during development.

Q7. What is Software Quality Control (SQC)?

Answer:
Software Quality Control (SQC) is the process of **checking whether the software product follows the quality plan** made during Software Quality Assurance (SQA).

It focuses on **reviewing the product** at different stages to find and fix problems before release.

Main Activities in SQC:

1. **Review of Requirements**
    - Checking if user needs are clear and complete.
2. **Design Reviews**
    - Making sure the software is designed properly and can be built correctly.
3. **Code Reviews**
    - Reading the source code to find bugs or bad practices.
4. **Testing the Product**
    - Running the software to catch any defects or failures.
5. **Deployment Review**
    - Ensuring the final version is ready and stable for use.

Goal of SQC:

- To **verify that the product is correct** and matches the original plans and user requirements.

Conclusion:

While **SQA focuses on improving the process**, **SQC focuses on checking the actual product**. Both are important for building high-quality software

Q8. Differentiate between Software Quality Assurance (SQA) and Software Quality Control (SQC).

Answer:

SQA	SQC
Software Quality Assurance	**Software Quality Control**
Focuses on the **process** used to create the software.	Focuses on the **product** that is being developed.
It is **preventive** — aims to **avoid defects** by improving the process.	It is **detective** — aims to **find defects** in the product.
Done **throughout the software development life cycle**.	Done **after or during product development** (e.g., during testing).
Activities include **planning, process monitoring, audits, reviews**.	Activities include **reviews, testing, inspections**.
Example: Ensuring developers follow coding standards.	Example: Testing the final software for bugs.

Conclusion:

- **SQA** ensures **quality is built into the process**.
- **SQC** ensures **quality is present in the final product**.
  Both are necessary for delivering high-quality software.