C1:

Task / Exercise C1 (Introductory) [20 mins for teams + 10 mins class discuss]:

1. What are the major tasks in developing and supporting a software system?.
2. What is the difference between a client and an end-user?
3. Discuss whether you think a programming language constraint may be viewed as a requirement. Explain why you think so.
4. Explain why developing a software **product** is a difficult task

1. Major Tasks in Developing and Supporting a Software System

1. **Understanding the Problem and Gathering Requirements**
2. **Planning and Design**
   - Includes resource management and technical solutions like UI/UX, database, and software architecture.
3. **Code/Develop the Software**
   - Implementation phase, including documentation.
4. **Test and Validate the Software**
   - Integration phase, launch, and making it available to the end user.
5. **Provide Support and Maintenance**
   - Continuous updating and bug fixing.

---

2. Difference between Client and End-User

1. **Client**: The individual or entity requesting and purchasing the software, who provides requirements.
2. **End-User**: The person who directly interacts with and uses the software.

---

3. Programming Language (PL) Constraints as a Requirement

- **Yes**, a programming language constraint can be seen as a requirement because:
  - Some languages may lack functionalities critical to development planning.
  - Clients may request a specific language, especially if they have existing systems or if they need compatibility with specific platforms (e.g., iOS or Android).

4. Reasons Developing a Software Product is Challenging

1. **Understanding Client Requirements and Their Changing Needs**
2. **Time and Budget Constraints**
3. **Selecting the Right Tools**
   - Choosing appropriate programming languages and frameworks.
4. **Teamwork and Human Factors**
   - Ensuring effective collaboration and selecting the right team members.
5. **Quality Assurance**
6. **Market Competition**
7. **Technical Decision-Making**
   - High-level architecture decisions.
8. **Maintaining Code Quality**
   - Organized programming with good practices and documentation.

C2:

`Task / Exercise C2 (Roles) [20 mins for teams + 10 mins class discuss ]:`

1. `List the possible roles or positions within a software team.`

2. `To which roles would you assign the following tasks?`
   a. `Change a subsystem interface to accommodate a new requirement.`
   b. `Communicate the subsystem interface change to other teams.`
   c. `Change the documentation as a result of the interface change.`
   d. `Design a test suite to find defects introduced by the change.`
   e. `Ensure that the change is completed on schedule.`

1. **List the possible roles or positions within a software team.**
   - **Project Manager**: Oversees project scope, budget, and schedule, ensuring alignment with objectives.
   - **Business Analyst**: Gathers customer information and requirements, defines the target audience.
   - **Software Architect**: Designs the system architecture, makes technical decisions.

- **Software Developers/Programmers**: Write the code, developing the solution according to specifications.
- **UX/UI Designers**: Focus on user experience and interface design to make the software intuitive.
- **Documentation Editor/Technical Writers**: Create and manage technical documentation.
- **Testers/QA Engineers**: Perform testing to validate software quality.
- **Security Engineer**: Manages data protection and software security.

2. **Assign roles to the following tasks:**
   - **Change a subsystem interface to accommodate a new requirement**: Software Architect, Developers.
   - **Communicate the subsystem interface change to other teams**: Project Manager, Business Analyst.
   - **Change the documentation as a result of the interface change**: Documentation Editor, Technical Writers.
   - **Design a test suite to find defects introduced by the change**: Testers, QA Engineers.
   - **Ensure that the change is completed on schedule**: Project Manager.

C3:

Task / Exercise C3 (Risks and Projects) [20 mins for teams + 10 mins class discuss] :

Within your team, discuss and answer the following questions,

1. Why does it take so long to get software finished?
2. Why are development costs so high?
3. Why can't we find all errors before we give the software to our customers?
4. Why do we spend so much time and effort maintaining existing programs?
5. Why do we continue to have difficulty in measuring progress as software is being developed and maintained?
6. In your opinion, what are the main reasons of failure when developing a software product

1. Why Does It Take So Long to Finish Software?

1. **Poor Planning and Management**

    - Insufficient scope management often leads to project failure.
2. **Dependence on External Teams**
    - Delays occur when relying on third-party deliverables.
3. **Frequent Changes in Requirements**
    - Increases complexity, lengthening project timelines.
4. **Underestimation of Budget and Time**
    - Inadequate forecasting can lead to budget overruns and delays.
5. **Team Conflicts**
    - Poor communication and unresolved conflicts within the team can slow progress.
6. **Poor Risk Management**
    - Identifying risks early is crucial to avoid project delays.
7. **Choice of Tools and Technologies**
    - Proper technology selection ensures compatibility and project efficiency.

2. Why Are Development Costs High?

1. **Paying Team Members and Developers**
    - Personnel costs can be significant.
2. **Costly Technologies and Tools**
    - Many tools and technologies required for development aren’t free.
3. **Expensive Data Collection**
    - Acquiring quality data can be a major expense.

3. Reasons for Software Project Failure

1. **Excessive Time for Market Readiness**
    - Late delivery may cause the product to miss market opportunities.
2. **High Development Costs**
    - Lack of cost control can make the project unsustainable.
3. **Frequent Bugs and Poor Maintenance**
    - Ongoing issues can frustrate users and damage the product’s reputation.
4. **Incompatibility with Existing Systems**
    - Lack of integration with target environments limits usability.
5. **Security Vulnerabilities**
    - Security lapses can lead to data breaches, affecting users and the company.

4. Key Guidelines for Software Project Success

1. **Anticipate Errors Early**
    - Integrate error-checking throughout development, not just in testing.
2. **Deliver Iterative Updates**
    - Divide the project into smaller deliverables to allow for fast feedback and adjustments.
3. **Focus on Market Needs**

    - Ensure the solution aligns with customer needs as they evolve.
4. **Promote Good Team Dynamics**
    - Facilitate collaboration and effective communication among team members.
5. **Choose the Right Technology**
    - Select tools and technologies that best suit the project's requirements.

Conclusion

By assigning appropriate roles, addressing challenges early, and following best practices, software teams can improve their chances of success. Effective planning, communication, and adaptability are essential for delivering high-quality software on time and within budget.