**NAME: EMMANUEL CLINTON ODHIAMBO MAY 2024**

**INTRODUCTION TO SOFTWARE ENGINEERING ASSIGNMENT 2**

**ANSWERS**

1. **Define Software Engineering:**

Software Engineering is the systematic approach to designing, developing, testing, maintaining, and evolving software systems. It employs engineering principles, methodologies, and practices to create reliable, efficient, scalable, and maintainable software products that meet user and stakeholder needs within time, budget, and resource constraints.

1. **Software Engineering vs. Traditional Programming:**

Software Engineering encompasses a broader set of activities beyond programming, including requirements analysis, design, testing, and maintenance, while traditional programming primarily focuses on coding. Software Engineering emphasizes systematic processes, team collaboration, and adherence to quality standards, whereas traditional programming may lack such structured approaches.

1. **Software Development Life Cycle (SDLC):**

**Phases**: The SDLC typically consists of phases like Planning, Requirements Gathering, Design, Implementation, Testing, Deployment, and Maintenance.

**Brief Description**: Each phase involves specific activities such as defining project goals, eliciting and documenting requirements, creating system designs, coding, testing for quality assurance, deploying the software, and maintaining it post-release.

**Agile vs. Waterfall Models**: Agile emphasizes iterative development, adaptive planning, and customer collaboration, suitable for projects with evolving requirements. Waterfall follows a linear sequence of phases, ideal for projects with well-defined and stable requirements.

1. **Requirements Engineering:**

Definition: Requirements engineering involves eliciting, analyzing, documenting, and managing software requirements throughout the software development lifecycle.

Importance: It ensures alignment between software functionality and user needs, reduces the risk of project failure, and serves as a foundation for the entire development process.

1. **Software Design Principles:**

Modularity: Modularity involves dividing software into smaller, independent modules with well-defined interfaces. It enhances maintainability by facilitating easier updates and scalability by enabling reuse of modules across different parts of the software system.

1. **Testing in Software Engineering:**

Levels: Unit testing verifies individual components, integration testing validates interactions between components, system testing evaluates the entire system's functionality, and acceptance testing ensures the software meets user requirements.

Importance: Testing helps identify defects early, ensures software quality, enhances reliability, and builds user confidence.

1. **Version Control Systems:**

Definition: Version control systems track changes to source code, enabling collaboration among developers, managing code versions, and facilitating code integration.

Examples: Git, SVN, and Mercurial are popular version control systems. They offer features like branching, merging, history tracking, and collaboration tools.

1. **Software Project Management:**

Role of Project Manager: The software project manager oversees planning, execution, and delivery of software projects, ensuring they meet quality, schedule, and budget objectives.

Responsibilities and Challenges: Key responsibilities include project planning, resource allocation, risk management, stakeholder communication, and progress tracking. Challenges include managing scope changes, resolving conflicts, and balancing competing priorities.

1. **Software Maintenance:**

Definition: Software maintenance involves modifying, updating, and enhancing software to address defects, accommodate changing requirements, and improve performance.

Types of Maintenance Activities: Corrective maintenance fixes defects, adaptive maintenance adjusts software to changes in the environment, perfective maintenance enhances functionality, and preventive maintenance reduces future issues.

Importance: Maintenance ensures software remains usable, reliable, and relevant over its lifecycle, maximizing its value to users and stakeholders.

1. **Ethical Considerations in Software Engineering:**

Ethical Issues: Software engineers may face ethical dilemmas related to privacy violations, security breaches, biased algorithms, intellectual property infringement, and societal impact.

Adherence to Ethical Standards: Software engineers can adhere to ethical standards by following professional codes of conduct, considering ethical implications in decision-making, seeking consensus on ethical dilemmas, and advocating for ethical practices within their organizations.