**Task (1):** What are the differences between unit testing, system testing, white box testing and black box testing?

Unit Testing, System Testing, White Box Testing, and Black Box Testing are different levels and types of software testing that serve distinct purposes in the software development life cycle.

* **Unit testing:** Unit testing is the type of software testing level in which each individual component of the software is tested. Unit testing is generally performed by the developers. Unit testing can’t be used for those systems which have a lot of interdependence between different modules. It does not allow for parallel testing.
* **System testing:** System testing is done to check whether the software or product meets the specified requirements or not. It’s done by both testers and developers. It contains system testing and Integration testing. It is done through more positive and negative test cases.
* [**Black Box Testing**](https://www.geeksforgeeks.org/software-engineering-black-box-testing/)**:** is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. Only the external design and structure are tested.
* [**White Box Testing**](https://www.geeksforgeeks.org/software-engineering-white-box-testing/)**:** is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Implementation and impact of the code are tested.

**Task (2):** What are code injection, Dependency injection and SQL injection?

Code injection, dependency injection, and SQL injection are terms related to software development and security, but they refer to different concepts. Here's an overview of each:

1. **Code Injection:** Code injection is a security vulnerability that occurs when an attacker is able to insert or "inject" malicious code into a computer program or script. This unauthorized code can then be executed by the application.

- Example: In web applications, code injection can occur if user input is not properly validated or sanitized. An attacker may input code as part of a user form, and if the application doesn't handle this input correctly, the injected code could be executed by the server.

2. **Dependency Injection:** Dependency injection is a design pattern used in software development to achieve inversion of control. In this pattern, the dependencies of a class (such as other objects or services) are injected into it rather than the class creating or managing its dependencies.

- Purpose: Dependency injection helps improve modularity, testability, and maintainability of code by reducing the coupling between classes. It makes it easier to replace or modify components without changing the core logic of the application.

- Example: Instead of a class creating an instance of another class it depends on, the dependency is injected from the outside. This can be done through constructor injection, method injection, or property injection.

3. **SQL Injection:** SQL injection is a type of code injection technique where an attacker inserts malicious SQL statements into input fields or parameters of a query. If the application does not properly validate or sanitize user input, the injected SQL code can be executed by the database.

- Example: In a login form, if the application uses user input directly in a SQL query without proper validation, an attacker could input something like `'; DROP TABLE Users; --` as their username, causing the SQL query to become malicious and potentially delete the entire Users table.

In summary, code injection is a general term for the injection of malicious code into a program, dependency injection is a design pattern for managing class dependencies, and SQL injection is a specific type of code injection that targets SQL databases by manipulating queries through user input. It's important for developers to be aware of these concepts and implement best practices to prevent security vulnerabilities in their applications.

**Task (3):** Write the time complexity of different data structures

**Worst case:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data structure** | **Access** | **Search** | **Insertion** | **Deletion** |
| **Array** | O (1) | O(N) | O(N) | O(N) |
| **Stack** | O(N) | O(N) | O (1) | O (1) |
| **Queue** | O(N) | O(N) | O (1) | O (1) |
| **Singly Linked list** | O(N) | O(N) | O(N) | O(N) |
| **Doubly Linked List** | O(N) | O(N) | O (1) | O (1) |
| **Hash Table** | O(N) | O(N) | O(N) | O(N) |
| **Binary Search Tree** | O(N) | O(N) | O(N) | O(N) |
| **AVL Tree** | O (log N) | O (log N) | O (log N) | O (log N) |
| **Binary Tree** | O(N) | O(N) | O(N) | O(N) |
| **Red Black Tree** | O (log N) | O (log N) | O (log N) | O (log N) |

**Task (4):** What is multimap is used for?

A multimap is a data structure or container in computer programming that is used to store multiple values associated with a single key. Unlike a regular map or dictionary, which typically allows only one value per key, a multimap allows multiple values to be associated with the same key. Each key-value pair in a multimap is unique, but different keys can have the same associated values.

The main use cases for a multimap include:

**Storing Duplicate Keys:** Multimaps are useful when you need to associate multiple values with the same key. In scenarios where duplicate keys are expected, a multimap provides a convenient way to organize and access these associations.

**Grouping Data:** Multimaps are often employed to group related data together. For example, if you are dealing with a dataset where each person can have multiple phone numbers or addresses, you might use a multimap to store this information, with each person's name as the key and their associated contact details as the values.

**Implementing a Many-to-One Relationship:** In database terms, a multimap can be thought of as representing a many-to-one relationship, where multiple entities are associated with a single entity. This is useful in scenarios where a single key can have multiple values, but each value corresponds to the same key.

**Handling Non-Unique Relationships:** In certain situations, a relationship between entities may not be unique. For instance, in a graph where multiple nodes can have the same parent, a multimap can be used to represent the parent-child relationships efficiently.

**Efficiently Managing Data:** Multimaps provide a convenient way to manage and organize data where a one-to-one mapping between keys and values is not sufficient. They can help simplify code and provide a clearer representation of the underlying relationships in the data.

**Task (5):** What is the data structure which we use hash map in it and what is it’s uses?