# CSE 464 – Homework 1

1. Review ISO 25010 quality framework for following questions

    a. Describe quality categories in your own works

    **Functional Suitability**: How well the program meets the requirements set forth for a program and whether the program outputs the proper results depending on various inputs or user interactions.

    **Performance Efficiency**: Refers to how well the system operates with the allocated resources while being tested under various scenarios.

    **Compatibility**: How well a system can interact or exchange information between other systems and whether it can still properly function while interacting or exchanging with other systems.

    **Usability**: Determines if a program can be operated by certain users while remaining efficient under various test cases.

    **Reliability**: How well the program can run under various conditions for a certain amount of time.

    **Security**: How well the program protects information such that only specific types of users or other systems can access various types of information.

    **Maintainability**: How well the program can be modified, changed, corrected, etc. while maintaining a high level of efficiency.

**Portability**: How efficient a system can be moved from one operating environment to another while maintaining a high level of effectiveness.

b. Measurable quality factors: ~~Time behavior~~, ~~adaptability~~, ~~recoverability~~, ~~Interoperability~~, ~~data integrity~~, ~~learnability~~, ~~useful help features~~, ~~availability~~, ~~modifiability~~, and ~~confidentiality~~.

  i. Group quality factors

  **Functional Suitability**: N/A

  **Performance Efficiency**: Time Behavior

  **Compatibility**: Interoperability

  **Usability**: Learnability, Useful Help Features

  **Reliability**: Availability

  **Security**: Data Integrity, Confidentiality

  **Maintainability**: Modifiability

  **Portability**: Adaptability

  ii. Briefly explain how to measure and test quality factors: Time Behavior, Confidentiality, Data integrity, and recoverability

  **Time Behavior**: We can easily test and measure time behavior by testing whether patients are able to retrieve information within 30 seconds. Simply time various patients on how long it takes to retrieve their information to determine if time behavior is meeting the required time.

  **Confidentiality**: The way you can test this quality factor is testing if only verified credentials are used to login into the healthcare system. Using various types of credentials determine if only verified users are able to get into the system to check if this quality factor is operating properly.

**Data Integrity**: To measure and test this quality factor we would need to determine if only authenticated and authorized users are able to view medical records. One way you can test this is by using either dummy accounts or real accounts of varying authorization to check if one the authorized accounts are able to access medical records.

**Recoverability**: This quality factor can be tested and measured by timing the amount of time it takes for the system to reboot following either a technical failure or a purposely induced failure.

2. Password Verification
   a. Identify equivalent partitions

   Password length:

   x > 8

   Password Complexity:

   Lower Case: x > 0

   Upper Case: x > 0

   Numbers: x > 0

   Special Characters: x > 0

   b. Develop test cases (Weak Normal Form – Classes of valid values of inputs)

| Test Case # | Partition Tested | Input(S) | Expected Output |
|---|---|---|---|
| 1 | Valid | Password0# | ‘Strong’ |
| 2 | Valid | HeLLo@10 | ‘Strong Enough’ |
| 3 | Valid | aSU!@#$%22 | ‘Strong’ |
| 4 | Valid | Test#@!11 | ‘Strong Enough’ |

3. Equivalent Partitions

   a. Use Java files

      i. Identify equivalent classes

         Strategy is to determine the outputs of the program and connect those outputs back to what combination of inputs is required to obtain that specific output.

         Classes:

            Two Roots: b^2 > 4ac

            One Root: b^2 = 4ac

            Zero Roots: b^2 < 4ac

      ii. Design Test Cases (Strong Robust – One from each class)

| Test Case # | Partition Tested | Input(s) | Expected Output |
|---|---|---|---|
| 1 | Valid | a = 5, b = 10, c = 5 | One Root (Root1 = -1, Root2 = -1) |
| 2 | Valid | a = 10, b = 5, c = 10 | Zero Roots (Root1 = -1, Root2 = -1) |
| 3 | Valid | a = 5, b = 15, c = -5 | Two Roots (Root1 = 0.33, Root2 = -3.33) |
| 4 | Invalid | a = “hello”, b = 10, c = 5 | Invalid Input |
| 5 | Invalid | a = 5, b = “CSE464”, c = 5 | Invalid Input |
| 6 | Invalid | a = 5, b = 10, c = “World” | Invalid Input |

   b. Submit Junit test program