Software Verification and validation

Name: Umair khan

ID#: 6949

Module: 6th Semester

Submitted To: Sir Zain Shaukat

Date: 25th/06/2020

Question#1

Attempt all of the MCQs each carry equal marks.

1. When should company stop the testing of a particular software?
b. It depends on the risks for the system being tested

2. White-Box Testing is also known as Clear box testing
c. Clear box testing

3. Validation refers to a different set of tasks ensures that the software that has been built is traceable to Customer Requirements.
c. Validation

4. System Testing verifies that all elements mesh properly and overall system functions/performance is achieved.
d. System Testing

5. What do you verify in White Box Testing?
d. All of the above.

6. Verification refers to the set of tasks that ensures the software correctly implements a specific function.
a. Verification

7. Who performs the Acceptance Testing?
b. End users

8. Which of the following is not a part of Performance Testing?
c. Measuring the LOC.

9. Which of the following can be found using Static Testing Techniques?
a. Defect

10. Testing of individual components by the developers are comes under which type of testing?
c. Unit testing

Question#2:

Explain Black Box testing and White Box testing in detail.

BLACK BOX TESTING is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In Blackbox Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Here are some steps followed to carry out any type of Black Box Testing.

- Initially, the requirements and specifications of the system are examined.

- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones -

- **Functional testing** - This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** - This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

WHITE BOX TESTING is testing a software solution's internal structure, design, and coding. It is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing is based on the inner workings of an application and revolves around internal testing.

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

The white box testing is divided into two basic steps

- The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.
- The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing trial, and error testing and the use of testing tools as we will explain further on in this article.

Types of White Box Testing

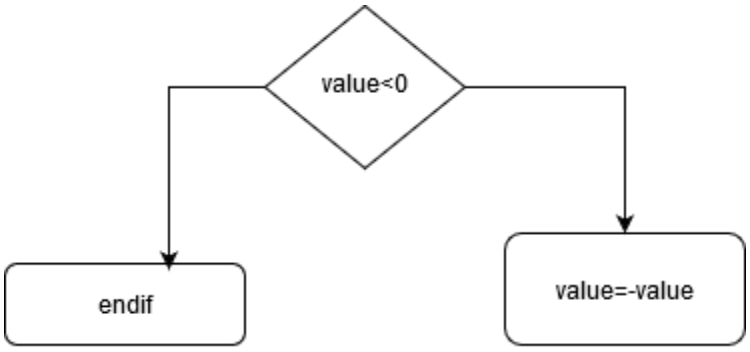
White box testing encompasses several testing types used to evaluate the usability of an application, block of code or specific software package. There are listed below --

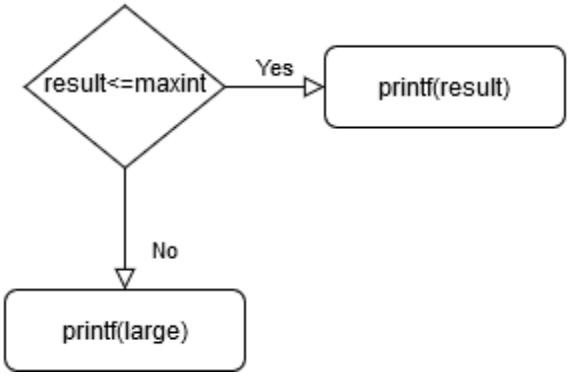
- Unit Testing: It is often the first type of testing done on an application. Unit testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.
- Testing for Memory Leaks: Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Q3:

CYCLOMATIC COMPLEXITY is a software metric used to measure the complexity of a program. It is a quantitative measure of independent paths in the source code of the program. Independent path is defined as a path that has at least one edge which has not been traversed before in any other paths. Cyclomatic complexity can be calculated with respect to functions, modules, methods or classes within a program.

This metric was developed by Thomas J. McCabe in 1976 and it is based on a control flow representation of the program. Control flow depicts a program as a graph which consists of Nodes and Edges.





- `Complexity = Edges - Nodes + 2*exit nodes`
- `Complexity = 8-6+2=4`

Question4. What is Z specification and why it is used for, also give some example this code written in Z specification?

The Z notation is a formal specification language used for describing and modelling computing systems. It is targeted at the clear specification of computer programs and computer-based systems in general

- Usage and notation

Z is based on the standard mathematical notation used in axiomatic set theory, lambda calculus, and first-order predicate logic. All expressions in Z notation are typed, thereby avoiding some of the paradoxes of naive set theory. Z contains a standardized catalogue (called the *mathematical toolkit*) of commonly used mathematical functions and predicates, defined using Z itself.

- Although Z notation (just like the APL language, long before it) uses many non-ASCII symbols, the specification includes suggestions for rendering the Z notation symbols

in ASCII and in Latex. There are also Unicode

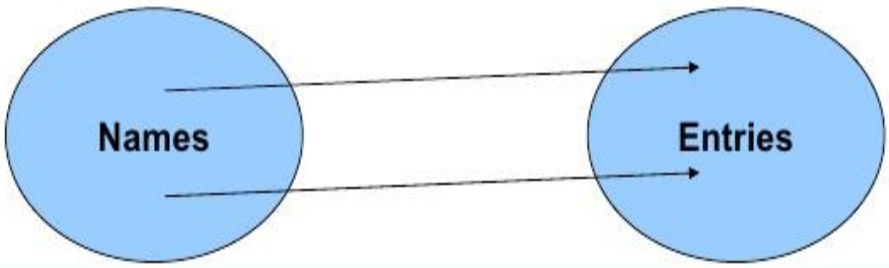
Data dictionary modeling

- A data dictionary may be thought of as a mapping from a name (the key) to a value (the description in the dictionary)
- Operations are
  - Add. Makes a new entry in the dictionary or replaces an existing entry
  - Lookup. Given a name, returns the description.
  - Delete. Deletes an entry from the dictionary
  - Replace. Replaces the information associated with an entry

Basic Data Representation

DataDictionary

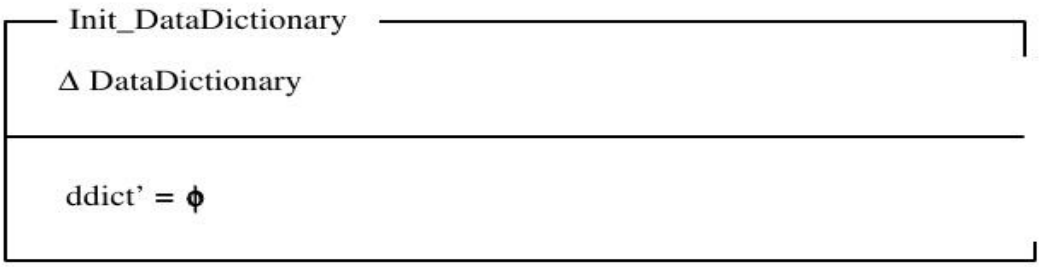
ddict: NAME ⇸ DataDictionaryEntry



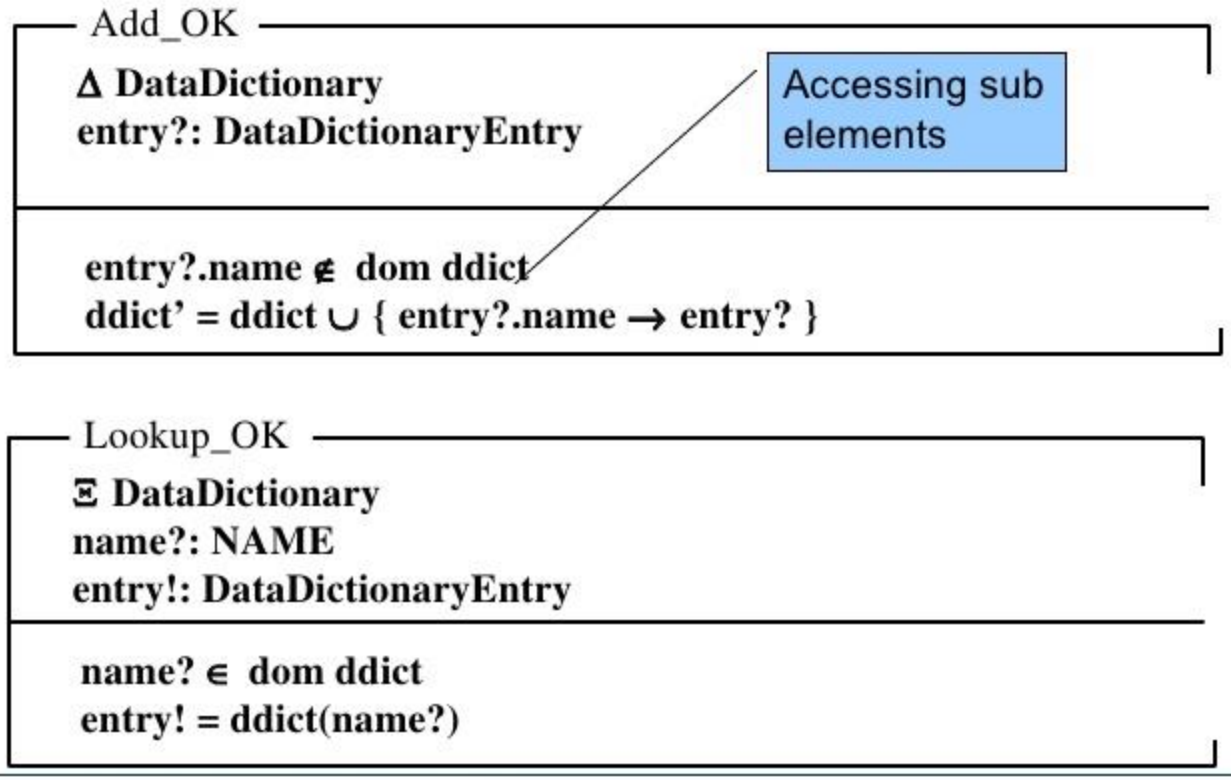
Function Summary

Name	Symbol	dom f	**One-to-one?**	ran f
Total function	$\rightarrow$	$= X$		$\subseteq Y$
Partial function	$\nrightarrow$	$\subseteq X$		$\subseteq Y$
Injection (total)	$\rightarrowtail$	$= X$	Yes	$\subseteq Y$
Surjection (total)	$\twoheadrightarrow$	$= X$		$= Y$
Bijection	$\rightarrowtail\!\!\!\!\twoheadrightarrow$	$= X$	Yes	$= Y$

Data dictionary initialization



Add and lookup operations



Add and lookup operations

Add_Error

Ξ DataDictionary
entry?: DataDictionaryEntry
error!: seq char

---

entry?.name $\in$ dom ddict
error! = "Name already in dictionary"

Lookup_Error

Ξ DataDictionary
name?: NAME
error!: seq char

---

name? $\notin$ dom ddict
error! = "Name not in dictionary"