Name: Danyal Ahmad
Class: BSSE
Course: Software Verification and Validation
ID # 13838
Module: Semester-VI
Submitted To: Zain Shaukat
SPRING-2020 Final Term ASSIGNMENT

Q1. MCQS (10)

1. When should company stop the testing of a particular software?

a. After system testing done
b. It depends on the risks for the system being tested
c. After smoke testing done
d. None of the above

2. White-Box Testing is also known as _____ .

a. Structural testing
b. Code-Based Testing
c. Clear box testing
d. All of the above

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

a. Verification
b. Requirement engineering
c. Validation
d. None of the above

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

a. Integration testing
b. Validation testing
c. Unit testing
d. System Testing

5. What do you verify in White Box Testing?
- Published on 03 Aug 15

a. Testing of each statement, object and function on an individual basis.
b. Expected output.
c. The flow of specific inputs through the code.
d. All of the above.

6. _____ refers to the set of tasks that ensures the software correctly implements a specific function.
- Published on 03 Aug 15

a. Verification
b. Validation
c. Modularity
d. None of the above.

7. Who performs the Acceptance Testing?
- Published on 03 Aug 15

a. Software Developer

b. End users
c. Testing team
d. Systems engineers

8. Which of the following is not a part of Performance Testing?
- Published on 30 Jul 15

a. Measuring Transaction Rate.
b. Measuring Response Time.
c. Measuring the LOC.
d. None of the above.

9. Which of the following can be found using Static Testing Techniques?
- Published on 29 Jul 15

a. Defect
b. Failure
c. Both A & B

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

- Published on 29 Jul 15

a. Integration testing
b. Validation testing
c. Unit testing
d. None of the above.

Q2. Explain Black Box testing and White Box testing in detail.

ANSWER(2):

BLACK BOX TESTING:

In Black-box testing, a tester doesn't have any information about the internal working of the software system. Black box testing is a high level of testing that focuses on the behavior of the software. It involves testing from an external or end-user perspective. Black box testing can be applied to virtually every level of software testing: unit, integration, system, and acceptance.

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:

White-box testing is a testing technique which checks the internal functioning of the system. In this method, testing is based on coverage of code statements, branches, paths or conditions. White-Box testing is considered as low-level testing. It is also called glass box, transparent box, clear box or code base testing. The white-box Testing method assumes that the path of the logic in a unit or program is known.

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.

KEY Difference b/w WHITE AND BLACK BOX TESTING:

- In Black Box, testing is done without the knowledge of the internal structure of program or application whereas in White Box, testing is done with knowledge of the internal structure of program.
- Black Box test doesn't require programming knowledge whereas the White Box test requires programming knowledge.
- Black Box testing has the main goal to test the behavior of the software whereas White Box testing has the main goal to test the internal operation of the system.
- Black Box testing is focused on external or end-user perspective whereas White Box testing is focused on code structure, conditions, paths and branches.

- Black Box test provides low granularity reports whereas the White Box test provides high granularity reports.
- Black Box testing is a not time-consuming process whereas White Box testing is a time-consuming process.

Q3. Find the cyclomatic Complexity and draw the Graph of this code.

Answer(3):

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.

In the graph, Nodes represent processing tasks while edges represent control flow between the nodes.

Program x:

Cyclomatic complexity of program X is the number of condition +1.

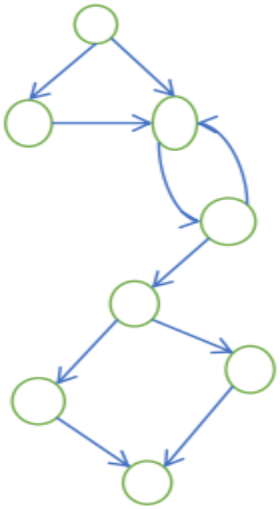
(Cyclomatic complexity = condition + 1)

There are two (2)

“if” conditions and 1 “while” condition.

Therefore program ‘X’ = 4

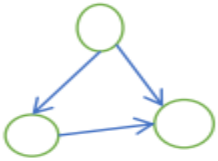
Control flow diagram program X:



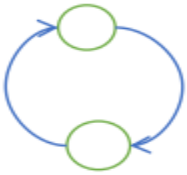
Edges=10

Vertices=8

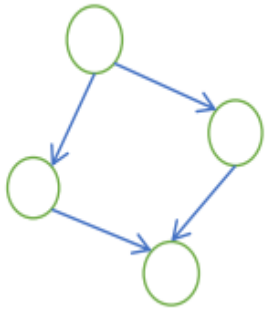
IF condition:



While condition:



IF Else condition:



Q4. What is Z specification and why its is used for, also give some example this code written in Z specification.

Answer(4):

Z Specification:

➔ Z Specification is a formal specification language

➔ used for describing and modeling computing systems

➔ based on the standard mathematical notation used in axiomatic set theory, lambda calculus and first-order predicate logic

With the ever-increasing complexity of computer systems, reliable and effective, design and development

of high quality systems that satisfy their requirements is extremely important. In the mission and safety critical system failure can cause cost overrun, loss of lives or even severe economic consequences can arise. So, in such situations, it is necessary that errors are uncovered before software is put into operation. These challenges call for acceptance of proper engineering methods and tools and have motivated the use of formal methods in software engineering. There are varieties of formal specification languages available to fulfill this goal and one way to achieve this goal is by using Z formal specification language. Z is model oriented formal method. based on set theory and first order predicate calculus.

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 catalog (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 encoding s for all standard Z symbols.

//Data dictionary entry

DataDictionaryEntry

entry: NAME

desc: seq char

type: Sem_model_types

creation_date: DATE

//Data dictionary as a function

DataDictionary

DataDictionaryEntry

ddict: NAME→ {DataDictionaryEntry}

//Data dictionary - initial state

Init-DataDictionary

DataDictionary

’ ddict’ = Ø

//Add and lookup operations

Add_OK

Δ DataDictionary

name?: NAME

entry?: DataDictionaryEntry

name? ∉ dom ddict

ddict’ = ddict ∪ {name? → entry?}

Lookup_OK

Ξ DataDictionary

name?: NAME

entry!: DataDictionaryEntry

name? ∈ dom ddict

entry! = ddict (name?)

Add_Error

Ξ DataDictionary

name?: NAME

error!: seq char

name? ∈dom ddict

error! = “Name already in dictionary”

Replace_OK

Δ DataDictionary

name?: NAME

entry?: DataDictionaryEntry

name? ∈dom ddict

ddict' ⊕{name? → entry?}

//Delete entry

Delete_OK

Δ DataDictionary

name?: NAME

name? ∈dom ddict

ddict' = {name?} ddict

Extract

DataDictionary

rep!: seq {DataDictionaryEntry}

in_type?: Sem_model_types

∀n : dom ddict • ddict(n). type = in_type? ⇒ddict (n) ∈rng rep!

∀i : 1 ≤ i ≤ #rep! • rep! (i).type = in_type?

∀i : 1 ≤ i ≤ #rep! • rep! (i) ∈rng ddict

∀i , j: dom rep! • (i < j) ⇒rep. name(i) < NAME rep.name (j)

THE END