NAME Ihsan ulllah

ID 6948

SUBJECT Software verification & Validation

Date 25/6/2020

QUESTION NO 1

. 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

- **b) it depends on the risk for the system being tested**

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

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

- **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

- **C) Validation**

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

- **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.

- **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.

- **Verification**

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

a. Software Developer
b. End users
c. Testing team
d. Systems engineers

- **b) End users**

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.

- **c) Measuring the LOC**

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

- **a) Defect**

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.

- **c) Unit testing**

QUESTION NO 2

. Explain Black Box testing and White Box testing in detail. (10)

ANS:-

Black Box testing:-

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 Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program

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.

Tools used for Black Box Testing:

Tools used for Black box testing largely depends on the type of black box testing you are doing.

For Functional/ Regression Tests you can use - QTP, Selenium

For Non-Functional Tests, you can use – Load Runner, Jmeter

Black Box Testing Techniques

Following are the prominent Test Strategy amongst the many used in Black box Testing

Equivalence Class Testing: It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.

Boundary Value Testing: Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.

Decision Table Testing: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

White Box Testing:-

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.

Following are important White Box Testing Techniques:

Statement Coverage

Decision Coverage

Branch Coverage

Condition Coverage

Multiple Condition Coverage

Finite State Machine Coverage

Path Coverage

Control flow testing

Data flow testing

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.

Apart from above, a few testing types are part of both black box and white box testing. They are listed as below

White Box Penetration Testing: In this testing, the tester/developer has full information of the application's source code, detailed network information, IP addresses involved and all server information the application runs on. The aim is to attack the code from several angles to expose security threats

White Box Mutation Testing: Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

White Box Testing Tools:-

Below is a list of top white box testing tools.

Parasoft Jtest

EclEmma

NUnit

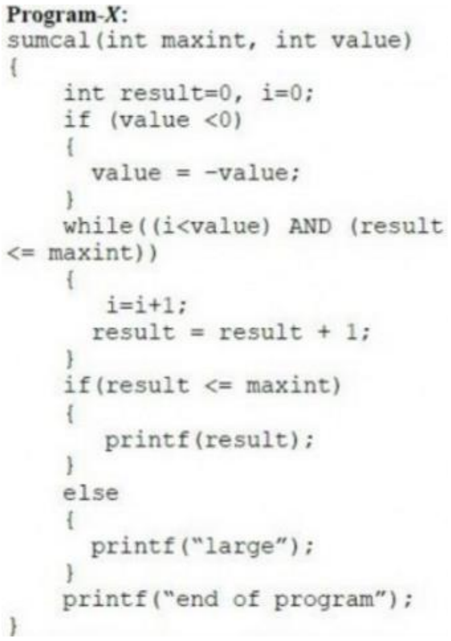
PyUnit

HTMLUnit

CppUnit

QUESTION NO 3

. Find the cyclomatic Complexity and draw the Graph of this code. (15)



Program-*X*:

```
sumcal(int maxint, int value)
{
    int result=0, i=0;
    if (value <0)
    {
      value = -value;
    }
    while((i<value) AND (result
<= maxint))
    {
        i=i+1;
      result = result + 1;
    }
    if(result <= maxint)
    {
       printf(result);
    }
    else
    {
      printf(“large”);
    }
    printf(“end of program”);
}
```

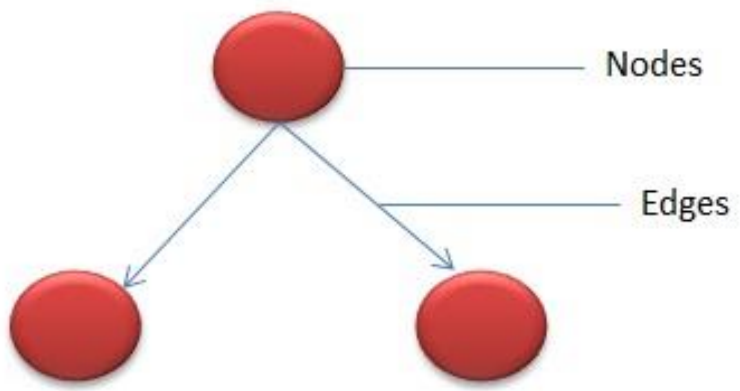
ANS:-

CYCLOMATIC COMPLEXITY:-

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.

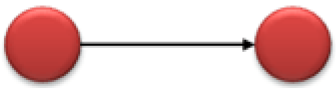
In the graph, Nodes represent processing tasks while edges represent control flow between the nodes.



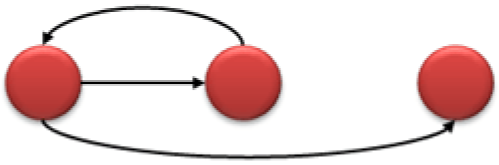
Flow graph notation for a program:

Flow Graph notation for a program defines several nodes connected through the edges. Below are Flow diagrams for statements like if-else, While, until and normal sequence of flow.

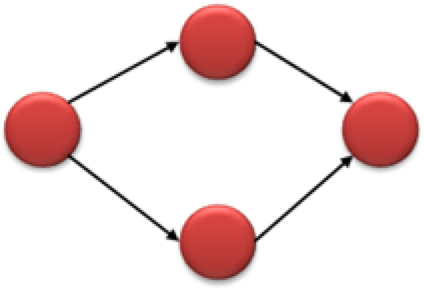
Sequence



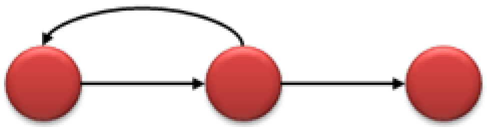
While



If-then-else



Until

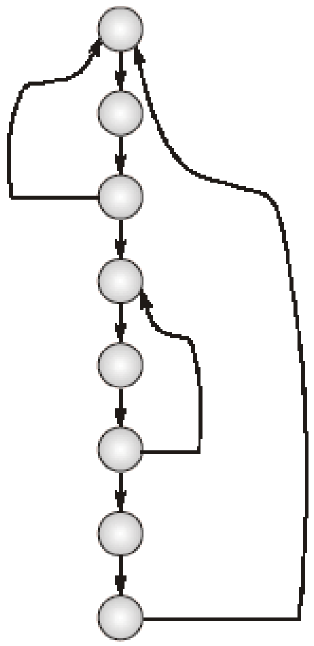


Consider three software items: Program-X, Control Flow Diagram of Program-Y and Control Flow Diagram of Program-Z as shown below

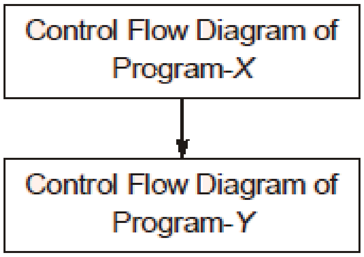
Program-*X*:

```
sumcal (int maxint, int value)
{
     int restult=0, i=0;
     if (value <0)
     {
        value = –value;
     }
     while ( (i < value) AND (result
< = maxint) )
     {
        i = i + 1;
     result = result + 1;
     }
     if (result <= maxint)
     {
         print f (result) ;
     }
     else
     {
         print f(“large”);
     print f (“end of program”);
}
```

Control Flow Diagram of Program-*Y*:



Control Flow Diagram of Program-*Z*:


Control Flow Diagram of Program-X
Control Flow Diagram of Program-Y

The values of McCabe’s Cyclomatic complexity of Program-X, Program-Y and Program-Z respectively are

(A) 4, 4, 7
(B) 3, 4, 7
(C) 4, 4, 8
(D) 4, 3, 8

Explanation:

The cyclomatic complexity of a structured program[a] is defined with reference to the control flow graph of the program, a directed graph containing the basic blocks of the program, with an edge between two basic blocks if control may pass from the first to the second. The complexity M is then defined as.

$$M = E − N + 2P,$$

where

E = the number of edges of the graph.

N = the number of nodes of the graph.

P = the number of connected components

QUESTION NO 4

. What is Z specification and why its is used for, also give some example this code written in Z specification. (15)

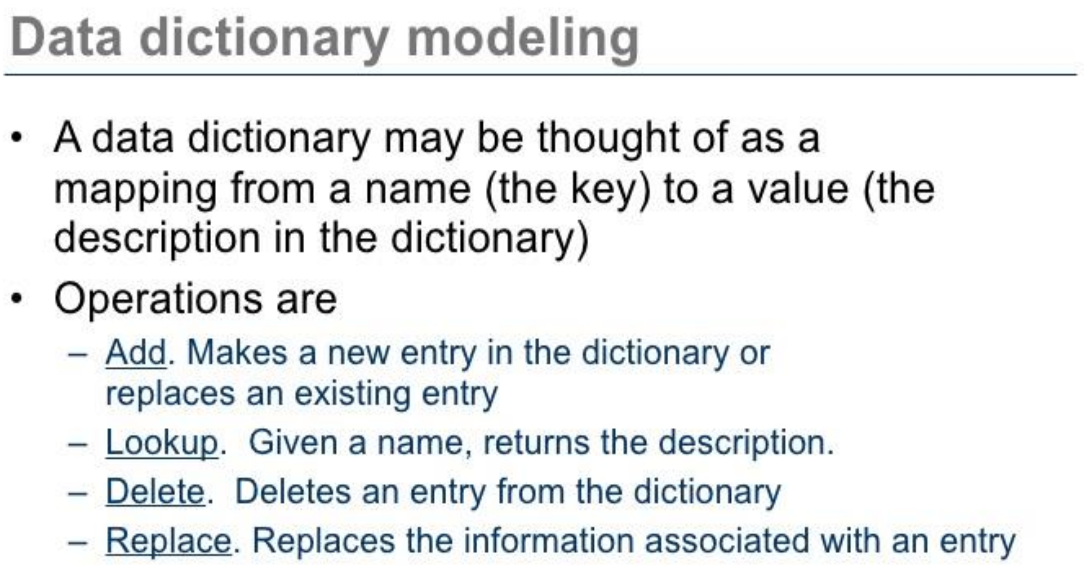
ANS:-

The **Z notation** is a formal specification language used for describing and modelling computing systems. It is targeted at the clear specification of computer programe 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



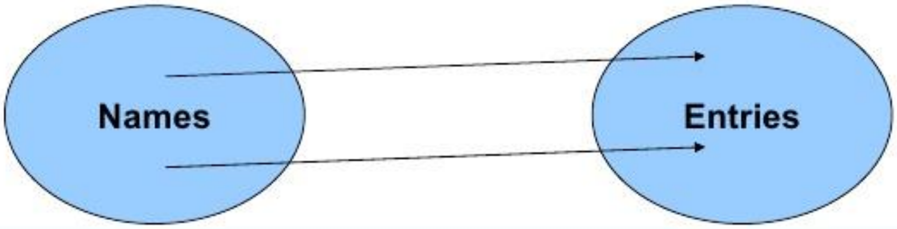
Data dictionary modeling

- 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	→	= X		⊆ Y
Partial function	⇸	⊆ X		⊆ Y
Injection (total)	↣	= X	Yes	⊆ Y
Surjection (total)	↠	= X		= Y
Bijection	⤖	= X	Yes	= Y

Data dictionary initialization

Init_DataDictionary

Δ DataDictionary

$ddict' = \phi$

Add and lookup operations

Add_OK

Δ DataDictionary
entry?: DataDictionaryEntry

---

entry?.name $\notin$ dom ddict
ddict’ = ddict $\cup$ { entry?.name $\rightarrow$ entry? }

Accessing sub elements

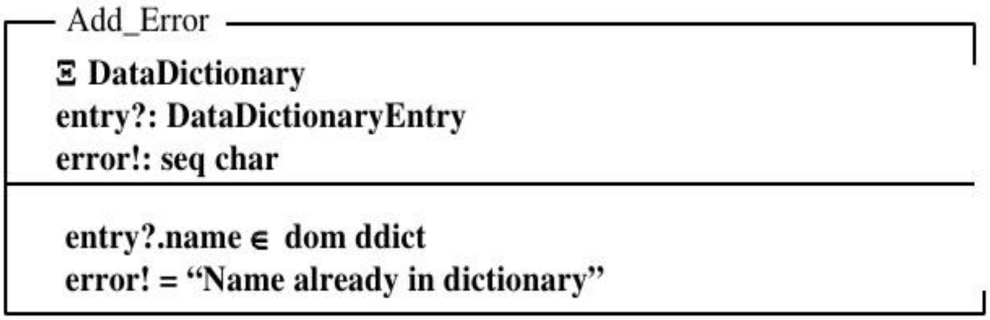
Lookup_OK

Ξ DataDictionary
name?: NAME
entry!: DataDictionaryEntry

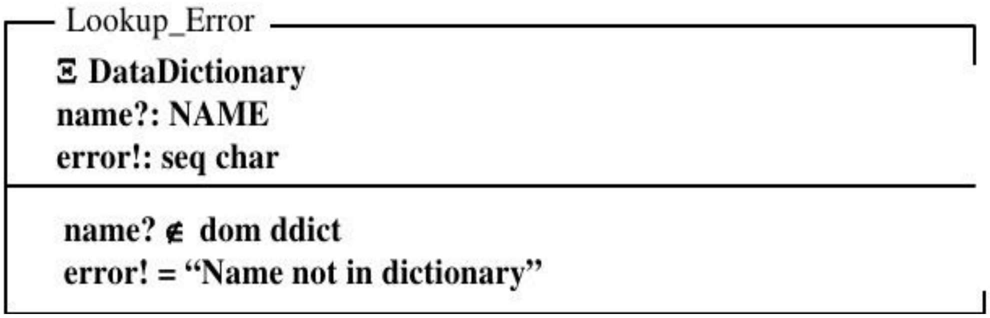
---

name? $\in$ dom ddict
entry! = ddict(name?)

Add and lookup operations



```
Add_Error
Ξ DataDictionary
entry?: DataDictionaryEntry
error!: seq char
------------------------------
entry?.name ∈ dom ddict
error! = “Name already in dictionary”
```



```
Lookup_Error
Ξ DataDictionary
name?: NAME
error!: seq char
------------------------------
name? ∉ dom ddict
error! = “Name not in dictionary”
```