NAME: HASSAN KAMAL

ID#: 12925

SUBJECT NAME: SOFTWARE VERIFICATION AND VALIDATION.

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
Answer:

It depends on the risks for the system being tested (b)

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

a. Structural testing
b. Code-Based Testing
c. Clear box testing
d. All of the above
Answer:
All of the above (d)
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
Answer:
Validation (c)
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
Answer:

System testing (d)

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
Answer:
All of the above (d)
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.
Answer:
Verification (a)

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

a. Software Developer
b. End users
c. Testing team
d. Systems engineers
Answer:
End users (b)

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.
Answer:
Measuring the LOC (c)
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
Answer:
Defect (a)

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.
Answer:
Unit testing (c)

Question#2
Explain Black Box testing and White Box testing in detail.
Answer:

Black Box Testing Techniques

There are three techniques usually employed by organizations and testers in case of Black Box Testing.

Equivalent Class Testing: It is used to reduce the number of possible test cases to an ideal level to maintain a reasonable test coverage.

Boundary Value Testing*:* It determines whether certain range of values are accepted by the software or not. This helps in reducing number of test cases.

Decision Table Testing: A decision table puts conditions and their outcomes in a matrix. There is a unique combination in every segment.

Advantages

- Suitable for large code segments
- Increased Efficiency
- Prior knowledge of code is not required

Black box testing is all about enhancing the user experience even if they are from a non-technical background. On the other hand, for technical support and precise coding, White box testing is an excellent approach for organizations to employ. Let's understand the nitty gritty of what goes behind White Box Testing.

White Box Testing

White Box Testing is also known as open, transparent or glass box testing. In white box testing, the tester has prior knowledge of the code and accordingly prepares the test case.

The tester has the knowledge of the internals of a system and knows how the system is implemented. The tester uses this knowledge to develop test cases that will examine the control flow, information flow, data flow, exception and error handling as well as coding practices of the system.

How does White Box Testing work?

Here's how White Box Testing works ...

1. The first step for the tester is to understand the source code.
2. White Box testing then involves testing of internal functions of the application, so knowledge of source code is crucial.
3. The tester should be aware of the secure coding practices as security is the most important factor in testing.
4. Tester can then write code for testing the application or can prepare certain test cases with suitable inputs.

White Box Testing

White Box Testing is also known as open, transparent or glass box testing. In white box testing, the tester has prior knowledge of the code and accordingly prepares the test case.

The tester has the knowledge of the internals of a system and knows how the system is implemented. The tester uses this knowledge to develop test cases that will examine the control flow, information flow, data flow, exception and error handling as well as coding practices of the system.

How does White Box Testing work?

Here's how White Box Testing works ...

1. The first step for the tester is to understand the source code.
2. White Box testing then involves testing of internal functions of the application, so knowledge of source code is crucial.
3. The tester should be aware of the secure coding practices as security is the most important factor in testing.
4. Tester can then write code for testing the application or can prepare certain test cases with suitable inputs.

White Box Testing Techniques

Code Coverage Analysis: It eliminates gaps in test case suite by identifying the program which cannot be examined by test cases. In addition, you can create test cases for untested part of the program which improves the quality of the software.

Statement Coverage: This technique checks every statement of the code at least once during the test cycle.

Branch Coverage: This technique tests every possible path in the code like If-else loops and other conditional loops of the software.

Advantages

- It optimizes the code as it tests every statement of the code.

- Automated testing is supported.
- Tests and test scripts can be reused.
- Testing is supported at early development stages.

Software testing is the most important part for maintaining the quality of the software. Manual and automated testing both are required to test the software thoroughly

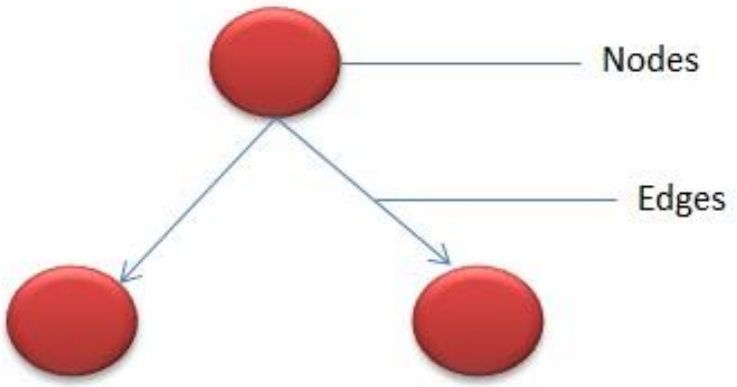
Q3. Find the cyclomatic Complexity and draw the Graph of this code.

Answer:

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.

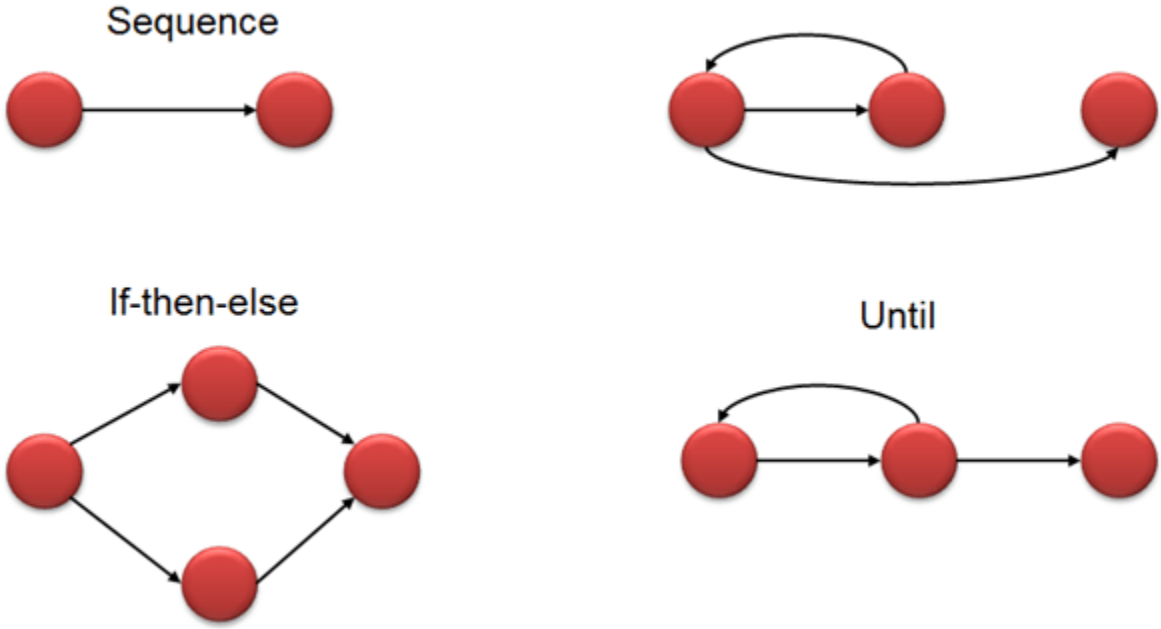
This metric was developed by Thomas J. McCabe in 1976 and it is based on a control flow representation of the program. Control flow depicts a program as a graph which consists of Nodes and Edges.

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



Flow graph notation for a program:

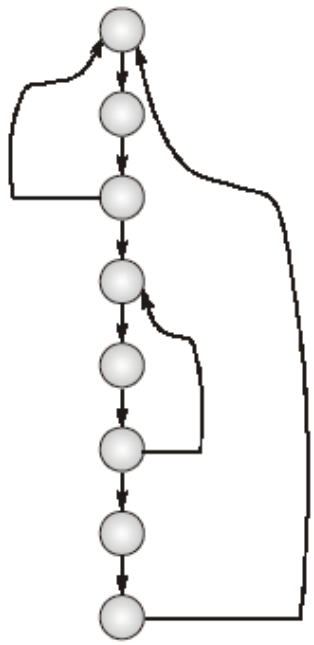
Flow Graph notation for a program defines several nodes connected through the edges. Below are Flow diagrams for statements like if-else, While, until and normal sequence of flow.



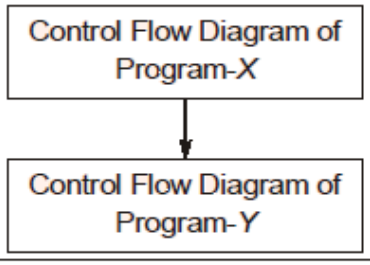
Program-*X*:

```
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*:



Question#4:

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

Answer:

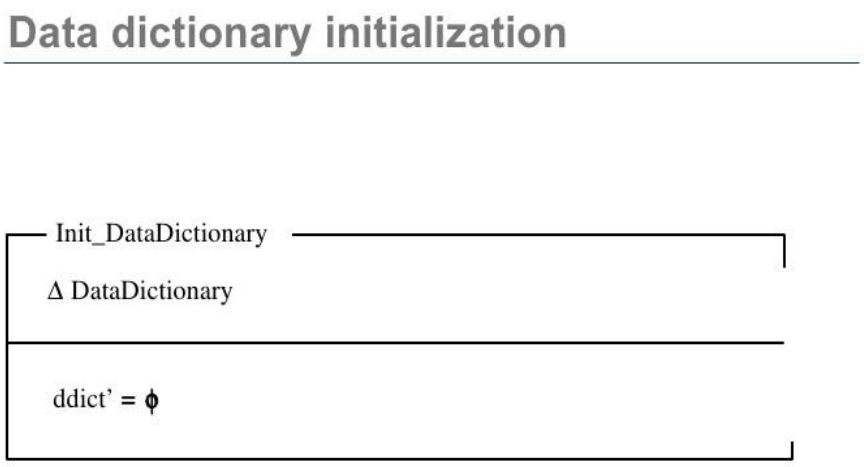
Definition:

Z specification is based on the typed of set theory. The most widely used formal specification language built upon schemas like basic building blocks, Allow modularity, Easier to understand by using graphical representation

Usage:

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 encodings for all standard Z symbols.



Add and lookup operations

Add_OK

Δ DataDictionary
entry?: DataDictionaryEntry

Accessing sub elements

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

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 $\in$ 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"