Final-Term – Semester Assignment

- Subject : Software Verification and Validation
- Submitted To : Sir Zain shaukat
- Submitted by : Mian Zabihullah
- Degree : BS(SE)
- ID # 6947
- Semester : 6th
- Date : 25/06/2020

Software Verification and validation

Marks: **50**

Q1. MCQS (10)

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

Ans: b. It depends on the risks for the system being tested

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

Ans: 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.

Ans: c. Validation

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

Ans: d. System Testing

5. What do you verify in White Box Testing?

Ans: d. All of the above.

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

Ans: a. Verification

7. Who performs the Acceptance Testing?

Ans: b. End users

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

Ans: c. Measuring the LOC.

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

Ans: a. Defect

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

Ans: c. Unit testing

☆ ☆ ☆

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

Ans: Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester

White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

Differences between Black Box Testing vs White Box Testing:

Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure r the code or the program of the software.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred as outer or external software testing.	It is the inner or the internal software testing.
It is functional test of the software.	It is structural test of the software.
This testing can be initiated on the basis of requirement specifications document.	It is mandatory to have knowledge of programming.

It is the behavior testing of the software.	It is the logic testing of the software.
It is the logic testing of the software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called as clear box testing.
It is least time consuming.	It is most time consuming.
It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: search something on google by using keywords	**Example:** by input to check and verify loops
Types of Black Box Testing: • A. Functional Testing • B. Non-functional testing • C. Regression Testing	**Types of White Box Testing:** • A. Path Testing • B. Loop Testing • C. Condition testing

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

Black Box Testing techniques:

○ Decision table testing ○ All-pairs testing ○ Equivalence partitioning ○ Boundary value analysis ○ Cause–effect graph ○ Error guessing ○ State transition testing ○ Use case testing ○ User story testing ○ Domain analysis ○ Syntax testing ○ Combining technique

☆ ☆ ☆

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

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 will be equal to four (4).

Formula

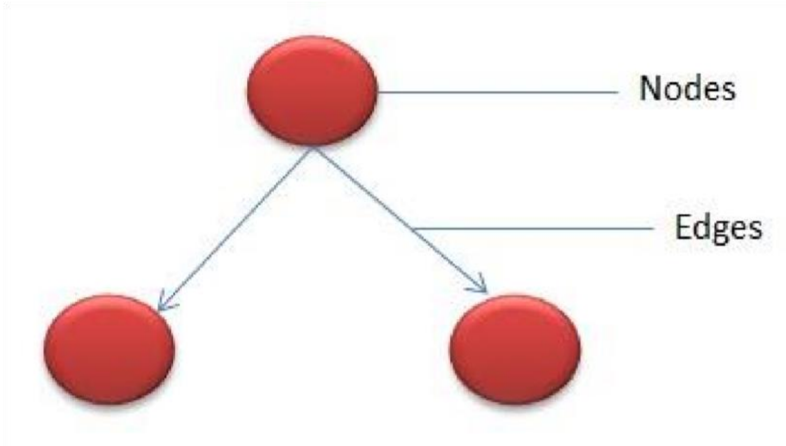
1. Cyclomatic complexity = No of predicates +1

For the Given program, predicates are if, while. Total 2 if and 1 while condition.

So answer will be 4.

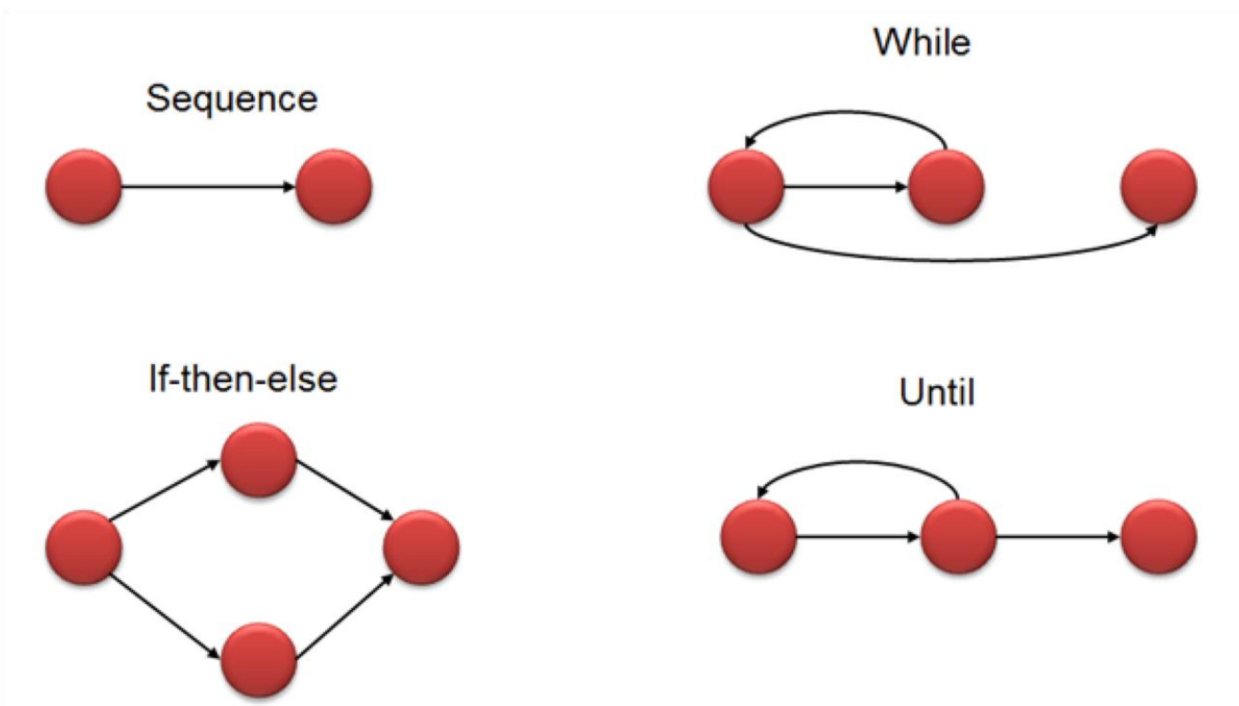
Graph of the Code:

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.

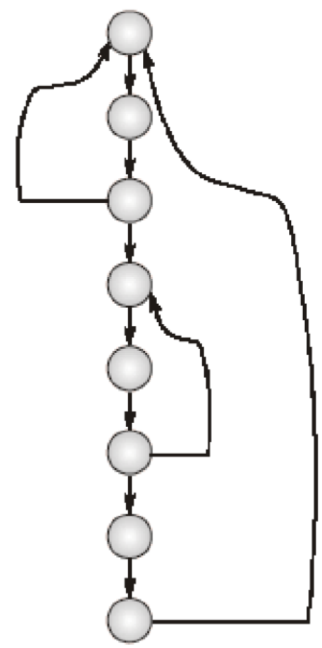


Consider three software items: Program-X, Control Flow Diagram of Program-Y and Control Flow Diagram of Program-Z as shown below

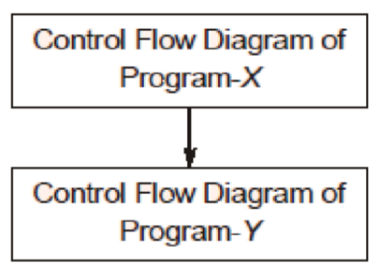
Program-*X*:

```
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*:



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

☆ ☆ ☆

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

Ans: Z Specification :

Z is a formal specification language for computer systems which is based on set theory and predicate logic. There are several textbooks on Z in the library, in particular: • The Mathematics of Software Construction. A. Norcliffe & G. Slater. Ellis Horwood, 1991. • Z User Manual. M.A. McMorran & J.E. Nicholls. IBM Technical Report, 1989. • The Z Notation - A Reference Manual. J.M. Spivey. Prentice–Hall, 1989. • An Introduction to Formal Specification and Z. B. Potter, J. Sinclair & D. Till. Prentice–Hall, 1996. The basic until of specification in Z is a schema. A Z schema consists of a name, a declaration of variables and a predicate.

: SchemaName x : X

Predicate

Here, variable x is declared to be of type X (see section 2.2). Note that the declaration part may declare more than one variable. The predicate part is a predicate (see section 2.3) whose free variables are those of the declaration plus any constants.

- **Why it is use for:**

The **Z notation** /ˈzɛd/ 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.

- **Example: 1**

Theater: Selling tickets

(no output variables in this schema)

Purchase()

TicketsForPerformance()

TicketsForPerformance()’

S? : Seat

P? : Person

S? & seating\ **dom sold**

Sold’ = sold ∪ {s? 7→ p?}

Seating’ = seating

 Example: 2

Response ::= okay | sorry

Success

r! : Response

---

r! = okay

Then

Purchase0 ∧ Success **is a schema that reports successful ticket sale**

Example: Data dictionary entry

[NAME, DATE]
sem_model_types = { relation, entity, attribute }

```
── DataDictionaryEntry ─────────────
name: NAME
type:
sem_model_types
creation_date: DATE
description : seq Char
────────────────────────────────────
#description ≤ 2000
────────────────────────────────────
```

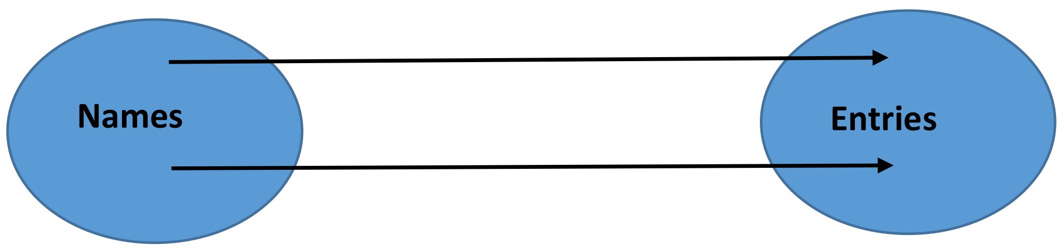
Data dictionary modeling

- 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 Intro | Hello World | Schema | Operations 24

Basic Data Representation:

Datadictionary

Ddict: NAME DataDictionaryEntry



Function Summary:

Name	Symbol	dom f	**One-to-one?**	ran f
Total function	→	$= X$		$\subseteq Y$
Partial function	⇸	$\subseteq X$		$\subseteq Y$
Injection (total)	↣	$= X$	Yes	$\subseteq Y$
Surjection (total)	↠	$= X$		$= Y$
Bijection	⤖	$= X$	Yes	$= Y$

Data dictionary initialization:

Init_datadictionary

DataDictionary

‘Ddict’ = 0

Add and lookup operations:

Add_OK

Δ DataDictionary

DataDictionaryEntry

Accessing sub elements

∉ dom ddict

ddict’ = ddict ∪ { entry?.name → entry? }

Lookup_OK

Ξ DataDictionary

name?: NAME

entry!: DataDictionaryEntry

---

name? ∈ dom ddict

entry! = ddict(name?)