**Software Testing**

**Assignment 1**

1) Explain why validation is more difficult than verification.

2) Differentiate between alpha and beta testing.

3) White box testing is complementary to Black Box testing, not alternative.

why ?Give an example to prove this statement.

4)Explain in Details the process to ensure the correctness of data flow in **given fragment of code?**

 Main()

 {

 Int k=35,z;

 Z=check(k)

 Printf(“/n%d”,z);

 }

 Check(m)

 {

 int m;

 If(m>40)

 return(1);

 Else

 return(0);

 }

5) What is the importance of “proof of correctness”.

6) What is the difference between a defect and a failure?

 7) Why is decision tables used in many cases rather than equivalence partitioning and boundary value analysis?

8) What is a ‘test plan’? What is a ‘test case’?

9) What steps are needed in developing and running software test?

10) How can you know when to stop testing?

**Assignment** **2**

1) Construct a test suit for the following fragment code using equivalence class testing method

 Main()

 {

 int runs;

 printf(“Enter the number of runs scored by India”);

 scanf(“%d”,&runs);

 if(runs<0)

 goto 53;

 else

 if(runs>200)

 {

 Printf(“India can compete with opponent team”)

 Exit();

 }

 S3:printf(“Invalid score Entered”);

 }

2) Discuss the pros and cons of Structural Testing.

3) Given the source code below:

 Void foo(int a,b,c,d,e)

 {

 If(a==0)

 Return;

 }

 Int x=0;

 If((a==b)or (c==d))

{

 X=1;

}

E=1/x;

}

List the test cases for statement coverage,branch coverage and condition coverage.

4) Write a C/C++ program to count the no.of characters,blanks,tabs in a line.Perform the following:

 a) Draw its flow graph

 b) Draw its DD-path Graph

 c) Find its V(G)

 d) Identify du-paths

 e) Identify dc-paths

5) Explain how risks are prioritized.

6) Can you explain equivalence partitioning?

7) Explain Top Down and Bottom Up approaches in testing?

8) Classify the following techniques into static or dynamic.

1. Equivalence Partitioning.
2. Use Case Testing.
3. Data Flow Analysis.
4. Exploratory Testing.
5. Decision Testing.
6. Inspections.

9) Find out the Cyclomatic complexity V (G) of the given flow graph.



10) List the pre-requisites for white-box testing.

**Assignment 3**

1) A->B,C

 B->D,E,F

 C->F,G

 F->H

 Give the Integration Testing for Bottom-up Testing

2) What do you understand by Big-Bang integration strategy.

3) “Staffing people for performance testing is most difficult”. Justify

4) Explain the following:

 a)Cause –Effect Graphing

 b)Mutation Testing

 c) Domain Testing

 d)Slice based testing

 e)Debugging

 5) Explain in brief the reduction schemes on prioritizing the test cases to reduce required testing effort ? As a tester, how do you evaluate and rank potential problems? Suggest some guidelines of your own approach to reduce number of test cases.

**6) Describe Code Review and Unit Testing Process.**

**7) Differentiate between Debugging and Unit Testing.**

**8) Describe bottom up approach for Integration Testing.**

9) Differentiate between exception and validation testing

10) What are the difference between Manual Testing and Automated Testing?

11) **Describe Code Review and Unit Testing Process.**

12) What is Smoke and scalability testing?

**Assignment 4**

1. Describe all the methods for class testing
2. Differentiate between static and dynamic testing tool.
3. What is the importance of technical reviews and why are technical reviews conducted ?
4. Explain the following :

a) Object oriented testing

b) Modern Testing Tools

1. Illustrate “How do you design interclass test cases”? What are the various testing methods applicable at class level?
2. **What is the main benefit of designing tests early in the life cycle?**
3. **What is the difference between Testing Techniques and Testing Tools?**
4. **What is random/monkey testing? When is it used?**
5. **Why are static testing and dynamic testing described as complementary?**
6. **What is negative and positive testing?**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***