Software Testing

Syllabus

Unit	Contents	Lectures
9.0	• **Software testing** • Introduction to Software Testing • Software Testing Needs and Goals • Testing Paths • Types of Testing Non – Execution Based Testing Execution Based Testing White Box Testing Black Box Testing	2

Syllabus

Unit	Contents	Lectures
9.1	• **Software testing** • Non execution based testing like walkthrough and review • Execution Based Testing • Tree Structure of Testing • Functional Testing Unit Testing Integration Testing	2

Syllabus

Unit	Contents	Lectures
9.2	• **Software testing** • System Testing Alpha Testing Beta testing Runtime Operations Testing • User Satisfaction Testing • Testing based on Matrix Testing	4

Introduction to Software Validation

- Software validation refers to the set of activities that ensures the software had been built according the customers requirement.
- It checks whether the right product is brought up or not.
- Software Validation encompasses a vide range of Software Quality Assurance (SQA) activities.
- They are
  - Formal Technical Review
  - Quality and Configuration Audits
  - Performance Monitoring

Introduction to Software Validation …

- Simulation
- Feasibility Study
- Documentation
- Database Reviews
- Algorithm with Analysis
- Development Testing
- Qualification Testing
- Installation Testing

Introduction to Software Testing

- Although testing plays a vital role in software validation, many other activities are also necessary.
- Software testing is one element of software validation.
- Testing provides a test cases from which quality can be assessed and more errors can be uncovered, but as we know that we cannot test quality, if quality is not there before we begin testing, it will not be there when we finish testing.

Introduction to Software Testing …

- Quality is incorporated into software throughout the process of software engineering. We can relate software testing to quality assurance by stating the underlying the motivation of software testing i.e. to affirm software quality with method that can be economically and mechanically applied to large scale system and small scale system.
- Software Testing is the critical elements of software quality assurance and represent the ultimate view of specifications, design and coding.

Features of Software Testing

- Testability
- Operability
- Observability
- Controllability
- Decomposability
- Simplicity
- Stability
- Understandability

Objective of Software Testing

- Testing is a process of executing the program with an intension of finding an error.
- A good test case is one that has high probability of finding an undiscovered error.
- A successful test is one that uncovered and undiscovered error.
- Testing is done successfully, if it will uncover errors.
- Testing cannot show absence of defects. It can only show software errors are presence.

Objective of Software Testing …

- Software is tested before the implementation to achieve
  - Leave no error for correction in maintenance.
  - To unearth potential problems in design and architecture in the early stage of development.
  - To ensure that the right software product is developed which is valid and verifiable.
  - To achieve user satisfaction and customer acceptance of the product.

Objective of Software Testing …

- The project team is committed to achieve desired quality and understands the implications.
- There exists a quality assurances infrastructure in the organization.
- Measures, measurement and metrics are documented for quality planning and as a guide to building effective testing strategies.
- Improvements in product, process and people are announced for the knowledge of and to encourage the software development team.

Basis of Software Testing

- Error

  Error is a discrepancy between actual value of the output and the specified correct value of the output.

- Fault

  Fault is the condition of the process in a software that results in malfunction. It is also termed as “bug”.

- Failure

  Failure is the inability of the software to perform a required function to its specification.

Approaches of Software Testing

- Top – Down Approach
- Bottom – UP Approach

What is to be Tested?

- Software products can be viewed as an assembly of different parts and components, each having specific role and objective to achieve.
- These parts and components are sub systems, modules, applications and unit level processes.
- Testing is done on systems, subsystems, modules, applications and unit processes to test whether
  - Requirements are met
  - Design is addressing the complete scope

What is to be Tested? …

- Architecture is efficient to implement the design
- There are no technology issues
- Modules are correctly designed to produce functionality
- Applications are correctly designed to produce results
- Processing is valid and verified for the correctness of business result
- In unit testing, basic code is tested

What to be Test Software?

- A software engineer must understand the basic principle that guide testing before designing test cases.
- The testing principles : -
  - All test should be traceable to customer's requirements.
  - Test should be planned long before testing begins.
  - Testing should begin in small and progress towards testing in large.
  - Extended testing is not possible.
  - To be most effective, testing should be conducted by third party.

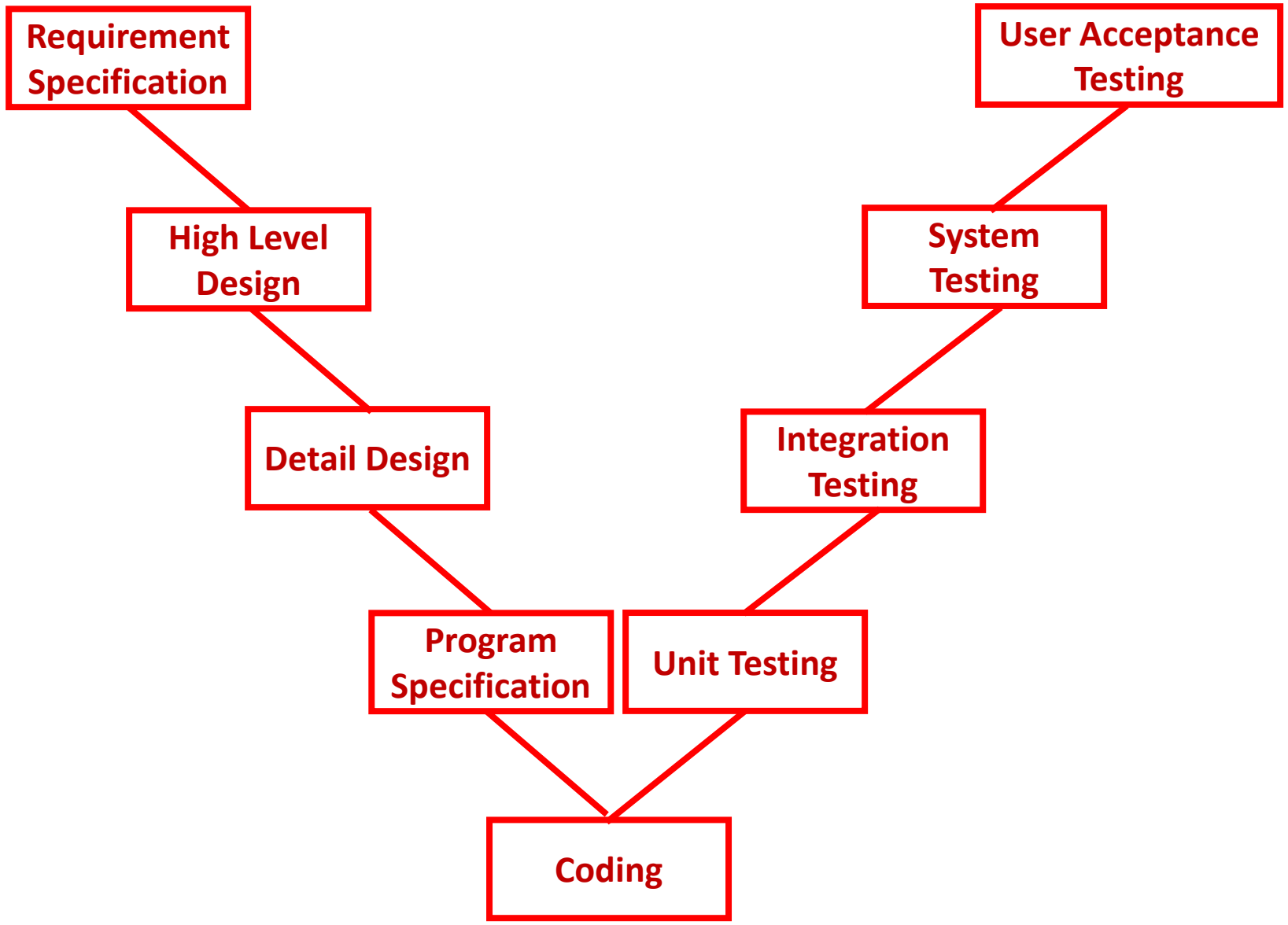
Validation Testing

- The process of evaluating software during the process of software development or at the end of the development process to determine whether it is satisfying the specified business requirements or not.
- Ensures that the software product actually meets need of the customer.
- Demonstrates that the software product fulfills the intended use when deployed on appropriate environment.

Validation Testing …

- Software Validation is done using
  - Unit Testing
  - Integration Testing
  - System Testing
  - User Acceptance Testing

Validation Testing …



Unit Testing

- Unit testing focuses on verification of smallest unit of software design i.e. module.
- Unit testing is normally unit box oriented.
- In unit test, module interface is tested to ensure that information flow is proper.
- Data Structure to ensure data integrity, boundary condition are tested to ensure module operate properly at boundary, independent paths and error paths are tested.
- Unit testing is done after coding takes place.

Unit Testing …

- Each test case is coupled with a set of expected results.
- Unit test is simplified with module with cohesion is observed.
- When only one function is addressed by a module, number of test cases is reduced, errors can be easily predicted are uncovered.
- Consider an input area code for a banking system.
- Area code may or may not be present. Range of values between say 100 and 999.
- This is also an example of **Equivalence Partitioning**.

Integration Testing

- It is a systematic technique for constructing a test to uncover errors associated with interfacing.
- It takes modules that have been unit tested and build up a program i.e. dictated by design.
- There are two types of integration
  - Non Incremental
  - Incremental

System Testing

- Testing of a fully integrated software system.
- Series of different types of tests with the purpose to exercise and examine the complete working of the integrated software system against the requirements.
- System testing includes the following
  - Verification of input functions to test whether it is producing the expected output or not.
  - Testing of integrated software by including external peripherals to check the interaction of various components with each other.

System Testing …

- Testing the whole system for end to end testing.
- Behavioral testing through user experience.

Types of System Testing

- Regression Testing
- Load Testing
- Functional Testing
- Recovery Testing
- Migration Testing
- Usability Testing
- Hardware Compatibility Testing

Types of System Testing: Regression Testing

- To confirm and identify that if there is any defect in the system due to the modification in any other part of the system.
- Ensures that the any changes done during the development process have not introduced any new defect.
- Gives assurance that old defects will not exist.

Types of System Testing: Load Testing

- Real time load testing is done.
- To clarify that whether the system can work under real time loads or not.

Types of System Testing: Functional Testing

- Finds whether there is any missing function in the system.
- A list of important functions is prepared and then checked whether they are there in the system or not. If not then can be added during the functional testing.
- Focus is on improvement of the quality of the system.

Types of System Testing: Recovery Testing

- Performed to confirm reliability, trustworthiness, accountability of the system.
- Should be able to recover from all possible system crashes successfully.

Types of System Testing: Migration Testing

- Checks whether the system if required can be modified as per new infrastructure with ease or not.

Types of System Testing: Usability Testing

- Ensures that the system is well familiar with the user and meets the required objectives.

Types of System Testing: Hardware Compatibility Testing

- Intends to check the compatibility of the software with the hardware.
- The hardware configuration must have to be compatible with the software to run it without any issue.

User Acceptance Testing

- The testing methodology that involves customers, stakeholders and end users in testing the software product to validate it against the requirement specification.
- Performed at the developer's site as well as customer's site.
- User acceptance testing is done using the
  - Alpha Testing
  - Beta Testing

Alpha (α) Testing

- Second Party or Third Party Testing.
- Second party testing means tested by the customer or user or representatives of the end users at the developer's side.
- Third party testing means tested by the independent team of testers.

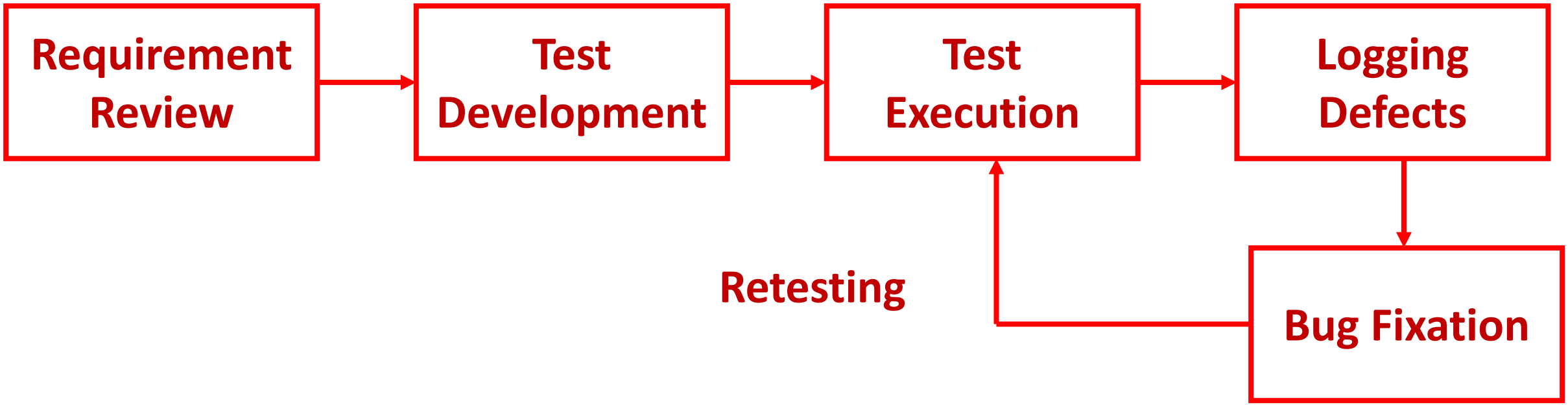
Nature of Alpha (α) Testing

- Simulated or real operational testing at an in – house site.
- Usually conducted once unit testing, integration testing, etc. is done.
- Depending on the requirement, can be white box or black box testing.

What is Alpha (α) Testing?

- Alpha testing follows following process
  - Requirement review
  - Test development
  - Test case design
  - Logging defects
  - Bug fixation
  - Retesting

What is Alpha (α) Testing? …



Phases of Alpha (α) Testing

- Alpha testing has two phases. They are
  - First phase
  - Second phase

Phases of Alpha (α) Testing: First Phase

- In – house developers of the software engineers do the first phase of alpha testing.
- In this phase, the tester used hardware debugger or hardware aided debugger to catch the bugs quietly.
- During the first phase of the alpha testing, the tester finds or tries to find the bugs, crashes, missing features and documents them.

Phases of Alpha (α) Testing: Second Phase

- In this phase, the main role of the tester is to ensure the quality assurance and keeping this goal in mind the tester(s) who can be the customer(s) or the user(s) or the representatives of the end user(s) or the team of independent tester(s) performs the testing by implementing white box testing or black box testing or both.
- In other words we can say that the main goal of the alpha testing is nothing but the user's acceptance.

When to Perform Alpha (α) Testing?

- As we know that alpha testing is nothing but the user acceptance testing.
- Alpha testing is performed once the (software) product has gone through stages of testing and prepared for the release.
- Alpha Testing is executed before Beta Testing.
- As we know that alpha testing is done at the developer's site under the monitoring of independent developer(s) or tester(s), who record the user experience and makes recommendations for necessary changes to enhance the user experience.

Why to Perform Alpha (α) Testing?

- Alpha testing is the final stage of the testing.
- Alpha testing is an essential and popular testing technique that helps the testing team to deliver quality and useful software product.
- Alpha testing is performed before the release of the software product.
- Alpha testing can define as the first round of independent testing that tries to ensure that the software runs as per the requirement document.

Why to Perform Alpha (α) Testing?

- Alpha testing is the final stage of the testing.
- Alpha testing is an essential and popular testing technique that helps the testing team to deliver quality and useful software product.
- Alpha testing is performed before the release of the software product.
- Alpha testing can define as the first round of independent testing that tries to ensure that the software runs as per the requirement document.

What Role Alpha (α) Testing does Play?

- Refines the software product by finding and rectifying bugs that were not discovered through previous tests.
- Alpha testing allows the team to test the software in the real world environment.
- Alpha testing helps to ensure the success of the software product.
- Alpha testing validates the quality, functionality of the software product and effectiveness of the software product before the release of the software product.

Features of Alpha (α) Testing

- Type of acceptance testing
- Alpha testing is happening at the stage of the completion of the software product.
- Alpha testing is conducted at the developer's site where the controlled real world environment is specified.
- Alpha testing us in – house testing, which is performed by the internal developers and testers within and outside the organization.
- Alpha testing is conducted to gain confidence in the user acceptance of the software product.

Features of Alpha (α) Testing …

- With the help of black box testing and white box testing technique, the goal(s) of alpha testing is achieved.
- Alpha testing tries to ensure the incorporation of maximum possible quality of software product prior to the beta testing.
- As the alpha testing is conducted at the developer's site (and by the developer in the first phase), enables the developer to record the error with ease to resolve found bugs promptly.

Features of Alpha (α) Testing …

- Alpha testing is done after the successful completion of unit level testing, integration level testing and system level testing.
- Alpha testing is for testing the software application, products and projects.

Advantages of Alpha (α) Testing

- Reduces the delivery time of the software product.
- Provides the complete test plan and test cases.
- Provides better observation of the reliability and accountability of the software.
- The feedback arrived helps to enhance the quality of the software.
- Frees the software developing team to concentrate in next software project.

Disadvantages of Alpha (α) Testing

- Alpha testing does not involve in in – depth testing of the software product.
- There are chances for the discrepancies due to the difference between the data for the tests to be conducted by the testers of the developer, testers of the customer and independent testers.
- Although the alpha testing is done in near real world environment at the developer site, real world environment components like multiple conditions, factors and circumstances fail the prevail there.

Beta (β) Testing

- A type of User Acceptance Testing.
- Beta testing is conducted at the end of the software testing life cycle.
- Beta testing is a type of field testing.
- Considered as external user acceptance testing.
- Conducted at the customer's site.
- Real users perform Beta Testing.

Beta (β) Testing …

- Beta testing is executed after Alpha Testing to check the accessibility, usability and functionality.
- Last phase of testing.

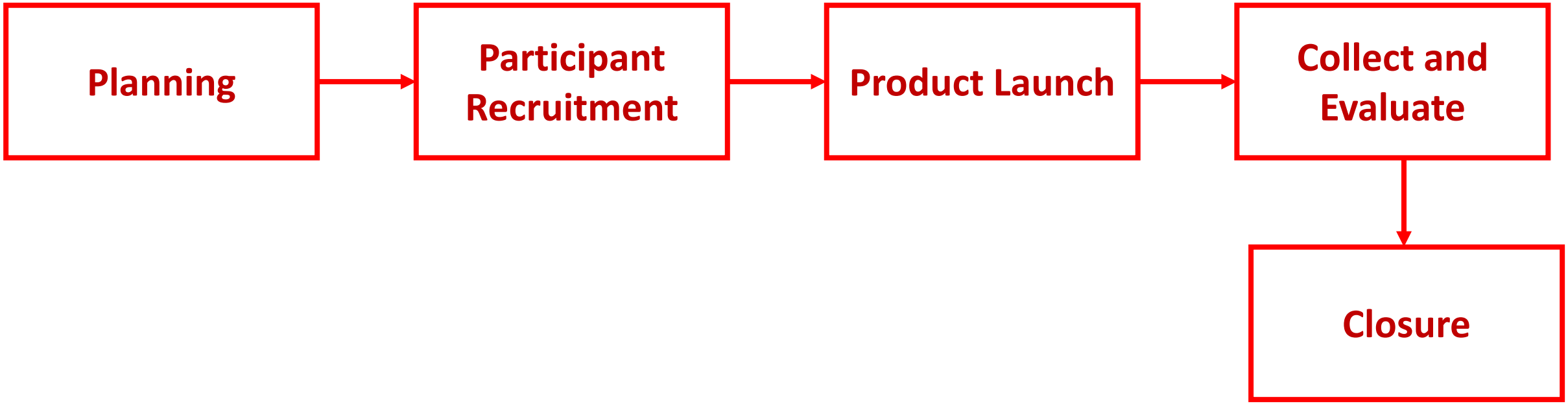
Nature of Beta (β) Testing

- Testing at an in – house site.
- Usually conducted once alpha testing is done.
- Conducted at the customer's site by the representatives of the actual users.

What is Beta (β) Testing?

- Beta testing follows following process
  - Planning
  - Participant Recruitment
  - Product Launch
  - Collect and Evaluate Feedback
  - Closure

What is Beta (β) Testing? …



When to Perform Beta (β) Testing?

- After Alpha testing.
- Before the (final) release of the software product.
- It means the product is about 90 – 95 % complete.

Why Beta (β) Testing is Performed?

- Beta Testing gives stress on customer satisfaction.
- Beta Testing helps to reduce the risk of product failure due to user validations.
- Beta Testing helps to get the direct feedback from end users.
- Beta testing helps to detect the defect and issues in the system that is either overlooked or remain undetected by the team of software testers.
- Beta testing helps the user to install, test and send actual feedback to the software developer.

Features of Beta (β) Testing

- Type of acceptance testing
- Beta testing is conducted in real world environment at the customer's site or user site.
- Beta testing helps is providing the actual position of the quality.
- Testing is conducted by the client, stakeholder and the end user.
- Beta testing is done after alpha testing and just prior to the release or launch of the software product.

Features of Beta (β) Testing …

- Beta testing is conducted in the absence of the tester and in the presence of real user.
- Beta testing is nothing but the black box testing.
- Beta testing is usually done for testing software products like utilities, operating system and application etc.
- Beta testing obtains direct feedback from the customer(s).
- Beta testing helps in testing the product in the customer's environment.

Types of Beta (β) Testing

- Open Beta Testing
- Closed Beta Testing
- Traditional Beta Testing
- Public Beta Testing
- Technical Beta Testing
- Focused Beta Testing
- Post release Beta Testing

Types of Beta (β) Testing: Open Beta Testing

- Software product is delivered to the users and the feedback from them is/are collected.
- Helps in getting feedback from diverse sources and enables the developer to further enhance the quality attributes.

Types of Beta (β) Testing: Closed Beta Testing

- Software product is delivered to the selected and limited users and the feedback from them is/are collected.
- Helps in getting feedback from diverse sources and enables the developer to further enhance the quality attributes.

Types of Beta (β) Testing: Traditional Beta Testing

- Software product is delivered to the market and the feedback from the users is/are collected.
- Helps in getting feedback from diverse sources and enables the developer to further enhance the quality attributes.

Types of Beta (β) Testing: Public Beta Testing

- Very much similar to open beta testing.
- Allows the software product to be released globally.
- Feedback and (evaluated) data is collected and based on those changes in requirement is done.

Types of Beta (β) Testing: Technical Beta Testing

- Involves delivering the software product to the initial groups of the organization.
- The data and feedback provided by the employees of the organization for the desired technical changes in future releases.

Types of Beta (β) Testing: Focused Beta Testing

- Focused on monitoring and evaluating a specific feature or component of the software.
- The software is released and the experience of the user is assessed and collected to make the required changes.

Types of Beta (β) Testing: Post Release Beta Testing

- Product is released to be used by the end users.
- The feedback, reactions and experience of end users are collected for the future release of the software.

Minimum Requirement for Beta (β) Testing

- Beta version of the software has to be ready.
- The environment should have to be ready to release the software application to the public.
- In order to capture the real time faults, the environment should have to be ready.
- All major and minor issues resolved.
- The feedback report should have to be prepared.
- The delivery of beta test summary report has to be done

Advantages of Beta (β) Testing

- Beta Testing gives stress on customer satisfaction.
- Beta Testing helps to reduce the risk of product failure due to user validations.
- Beta Testing helps to get the direct feedback from end users.
- Beta testing helps to detect the defect and issues in the system that is either overlooked or remain undetected by the team of software testers.
- Beta testing helps the user to install, test and send actual feedback to the software developer.

Disadvantages of Beta (β) Testing

- As beta testing is done at the customer's site and is conducted by the end user's in the real environment, the software engineer has no control over the process of the testing.
- As the end user's at the customer side has no experience in testing the software product, beta testing can be time consuming and may delay the final release of the software product.
- Beta testing does not test the functionality of the software product in depth.

Disadvantages of Beta (β) Testing …

- To work on the feedback of those ones who are not experienced (and actually not going to use the software) in what they are trying to perform(beta testing), it is going to be wastage of time and money.

Software or System Path Testing

- Non – Execution Based Testing
- Execution Based Testing

Non – Execution Based Testing

- The module is always reviewed by a team.
- The testing relies on fault detection strategy.
- The fault detecting power of non execution based testing techniques leads to rapid, thorough and early fault detection.
- Also termed as static testing or static program analysis.
- There are following types of non Execution Based Testing. They are
  - Walkthrough
  - Review

Non – Execution Based Testing: Walkthrough

- Walkthrough or Walk – Through
- A form software peer review.
- Designer or programmer leads member of the development team and interested parties to go through a software product and the participants ask questions and tender comments about possible errors, violations etc.

Walkthrough Objectives

- To gain feedback about the technical quality or content of the document.
- To familiarize the audience with the content.

Walkthrough Participants

- **Author**

  The one who presents the software product in step by step manner at the walkthrough meeting.

- **Walkthrough Leader**

  The one who conducts the walkthrough, handles administrative tasks and ensures orderly conduct.

- **Recorder**

  The one who notes all possible defects, decisions and action items identified during the walkthrough meetings.

Review

- Also termed as software technical review.
- Focuses on technical quality of the product.
- Form of peer review conducted by a team of qualified personnel.
- Examines the suitability of the software product for its intended use and tries to identify discrepancies from specifications and standards.
- Provides recommendations of alternatives and their comparative study.

Execution Based Testing

- The modules are run against test cases.
- Also termed as dynamic testing.
- There are two types of Execution Based Testing. They are
  - White Box Testing
  - Black Box Testing

White Box Testing

- White Box Testing involves knowledge which is related to the internal working of a product, tests are conducted to check internal operations as per specifications.
- It is also known as "**Glass Box Testing**".
- It is a test case design philosophy that uses the control structure described as part of component level design to derive test cases.
- With the help of White Box Testing, the software engineer can derive test cases that

White Box Testing …

- Guarantee that all independent paths within a module have been exercised.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to ensure their validity.

Black Box Testing

- Black Box Testing involves knowledge which is related to program functionality, it checks whether each function is fully operational.
- It is also called "**Behavioral Testing**".
- It focuses on functional requirement of statement.
- It enables software engineer to derive set of input condition that will fully exercise all functional requirement for a program.

Black Box Testing

- Black Box test attempts to find the following
  - Interface Error
  - Incorrect, missing functions
  - Data structure errors
  - Performance errors
  - Initialization and termination errors

How to Perform White Box Testing?

- Basis Path Testing
- Control Structure Testing

Basis Path Testing

- Basis Path Testing is a white box testing technique proposed by Tom McCabe.
- It enables the test case designer to deliver a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths.
- Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during the test.

Basis Path Testing …

- It involves four stages.
- They are
  - Flow Graph Notation
  - Independent Program Path
  - Deriving Test Cases
  - Graph Metrics

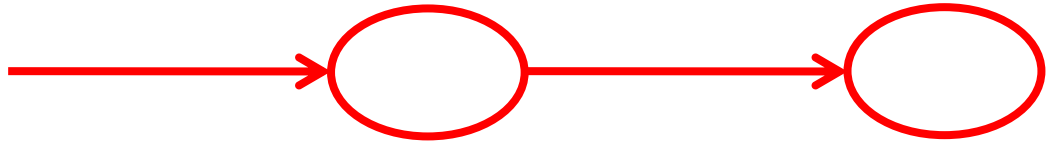
Flow Graph Notation

- A flow graph depicts logical control flow.
- Flow chart is used to depict program control structures, which is converted into a flow graph assuming that no compound conditions are contained in the decision diamonds of the flow chart.
- Each circle represents a flow graph node, which in turn represents one or more procedural statement.
- A sequence of process boxes and decision diamonds can have a single box or node.

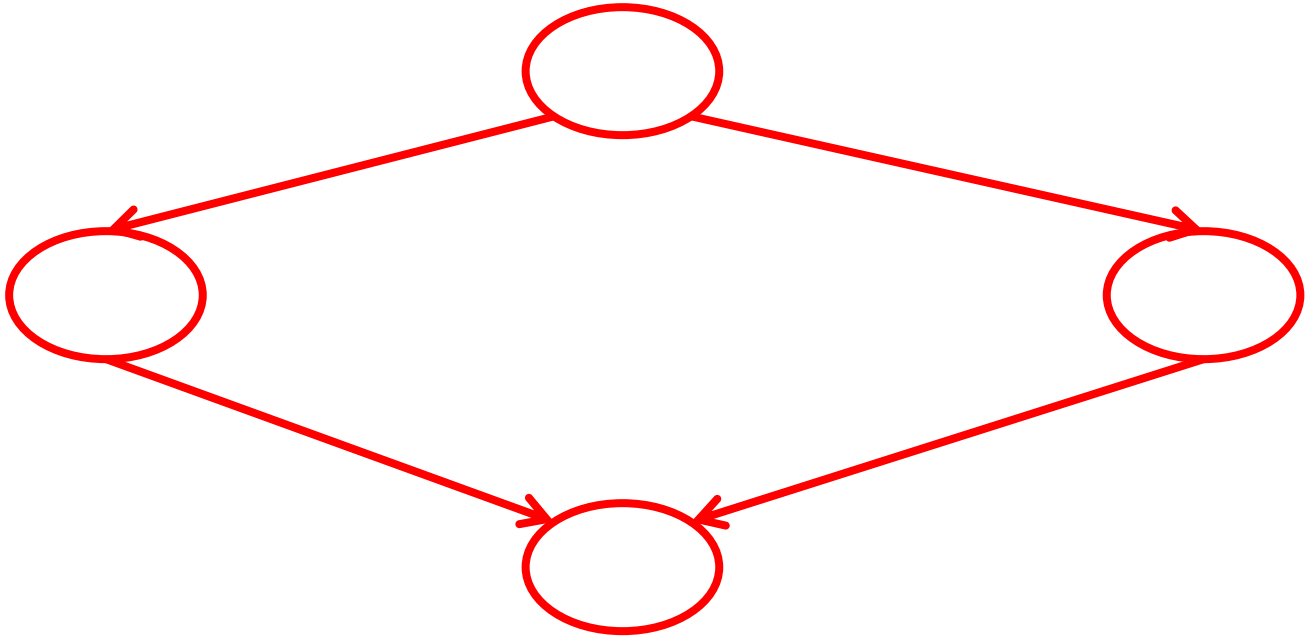
Flow Graph Notation …

- The arrows in the flow graph are called edges or links, which represent flow of control and analogues to flow chart arrows.
- An edge must terminate at a node even if a node does not represent any procedural statement.
- Areas bounded by edges or nodes are called regions.
- Types of procedural statements are

Flow Graph Notation …

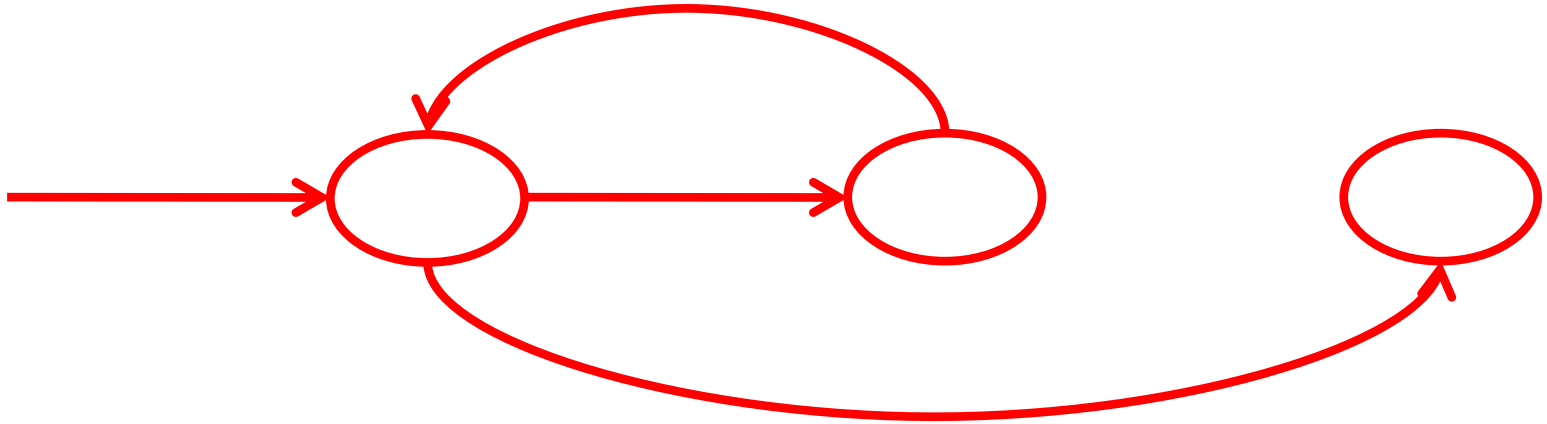


Represents “Sequences”

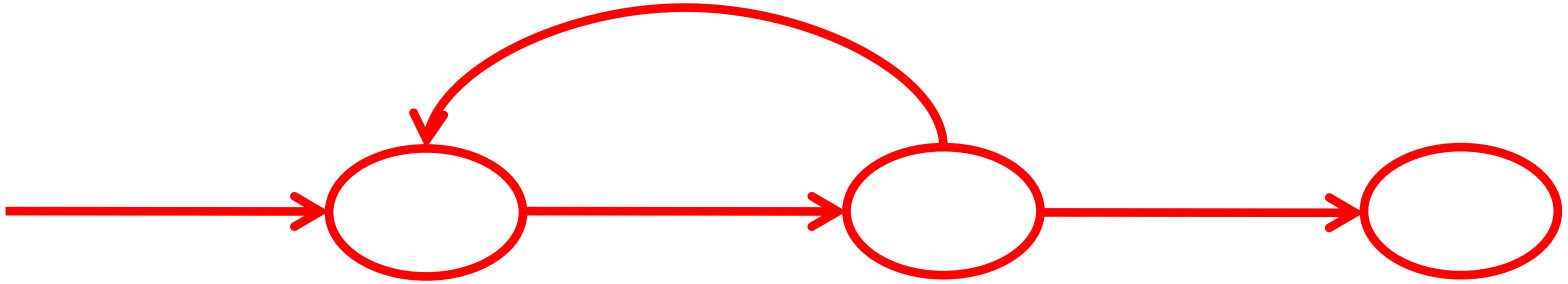


Represents “if-else” statement

Flow Graph Notation …

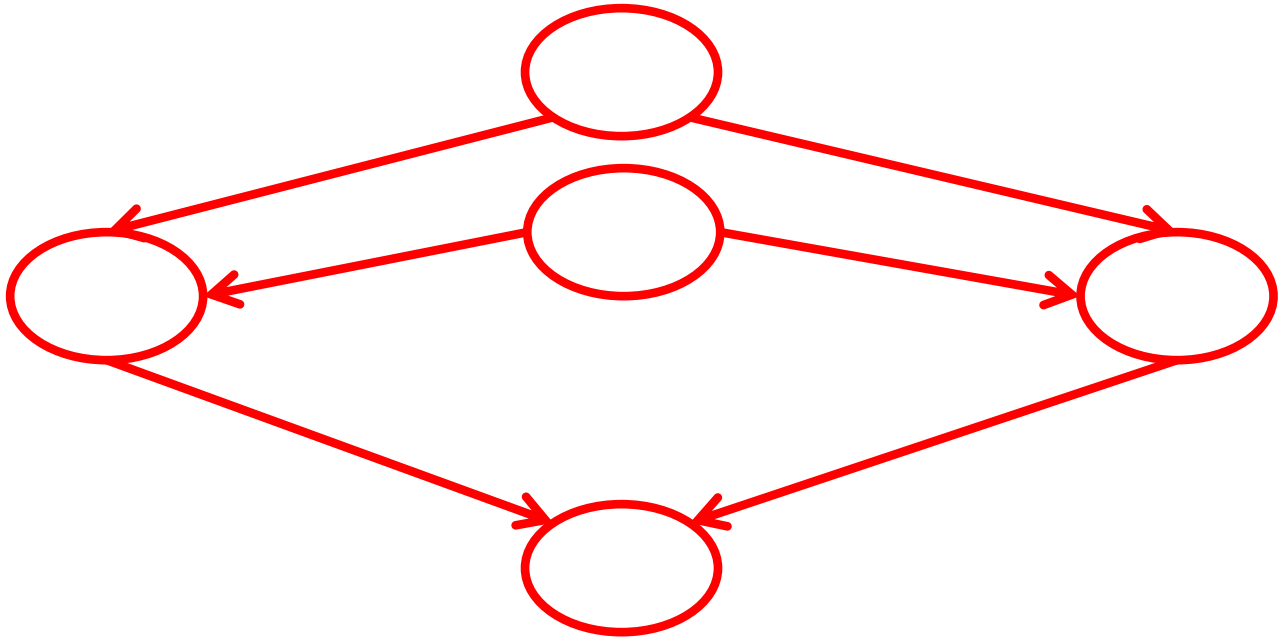


Represents “while” loop



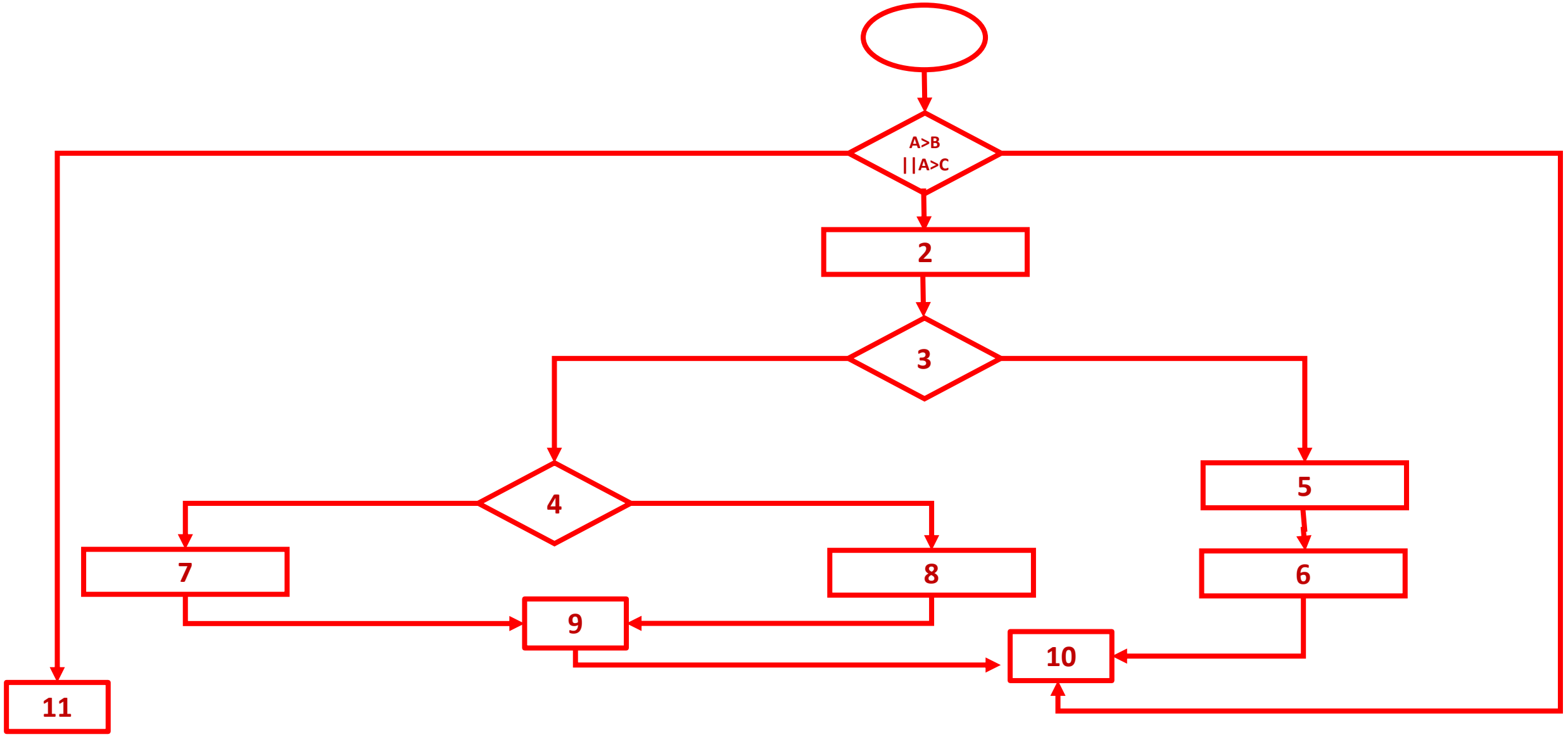
Represents “until”

Flow Graph Notation …

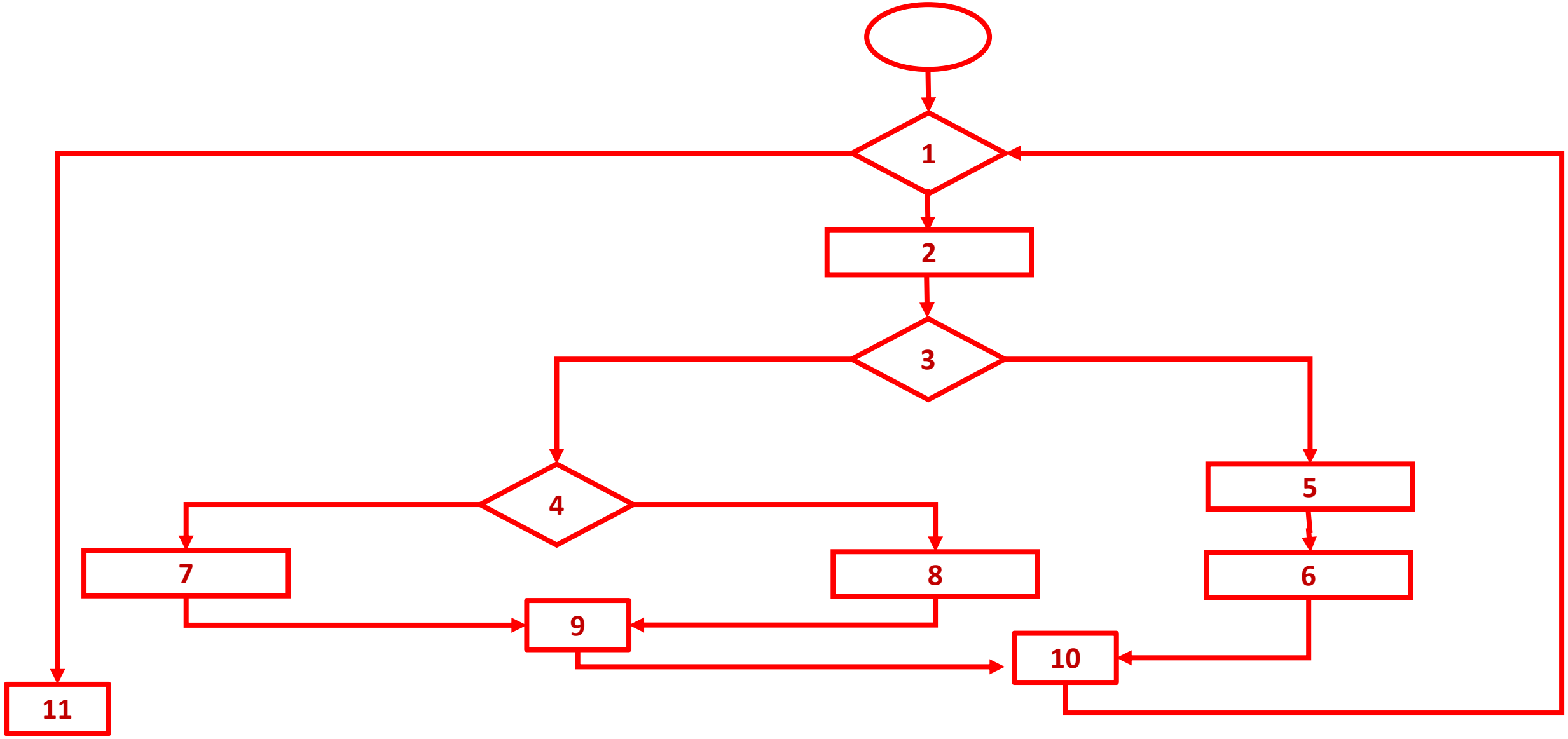


Represents “case” statement

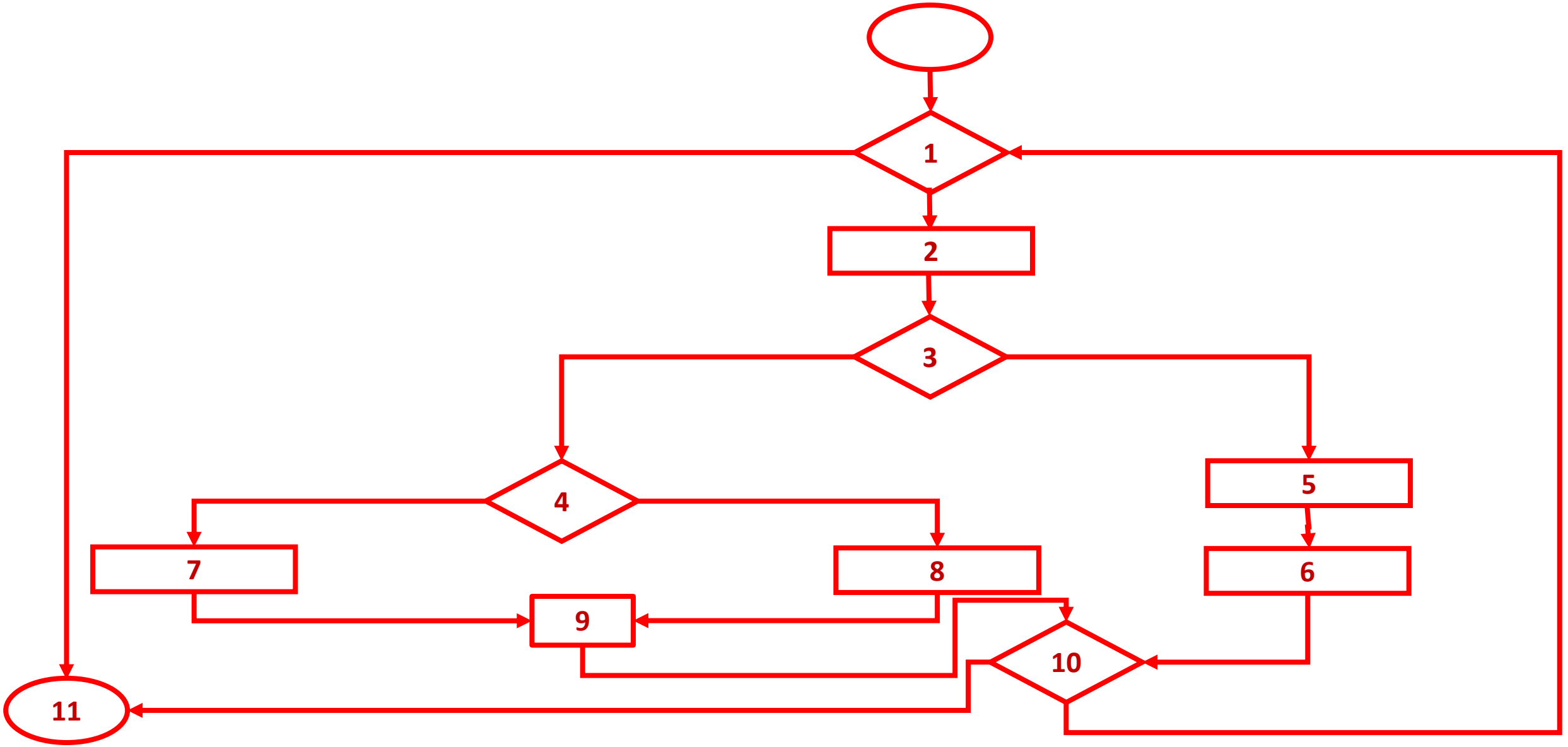
Flow Graph Notation …



Flow Graph Notation …



Flow Graph Notation …



Flow Graph Notation …

- Each node that contains a condition is called “Predicate Node”.
- It is characterized by two or more edges coming out from it.

Independent Program Path

- An independent program path is any path through a program that introduces at least one new set of processing statements or a new condition.
- When stated in terms of flow graph, an independent path must now along at least one edge that has not been traversed before the path is defined.
  - Path 1: 1- 11
  - Path 2: 1 -10
  - Path 3: 1 – 2 – 3 – 4 – 7 – 9 – 10 – 11
  - Path 4: 1-2-3-4-8-9-10 and Path 5: 1 – 2 – 3 – 5 – 6 – 10

Independent Program Path …

- Each path introduces a new edge and node.
- The path 1-2-3-4-5-10-1-2-3-6-8-10 is not an independent path because it is simply a combination of already specified path and does not traverse any new edge.
- Path 1-2-3-4 constitute a basis set for the program i.e. we can design test to face the executions of these path.
- Every statement in the program will have been guaranteed to be executed at least once and every condition will have been executed on its true and false sides.

Cyclomatic Complexity

- It is a software matrix that provides the quantitative measure of logical complexity of a program.
- Value computed for cyclomatic complexity defines number of independent paths in the basis sets.
- Cyclomatic complexity is computed in one of the three ways.
- The number of regions of flow graph corresponds to the cyclomatic complexity.
- Let V(G) = 4.

Derive Test Cases

- We can apply basis path testing in procedural design or source code.
- It is applied using following steps
  - Using design or code, draw corresponding program.
  - Determine Cyclomatic Complexities of the resultant flow graph.
  - Determine basis set of linearly independent path.
  - Determine test cases that force execution of each path in basis set.

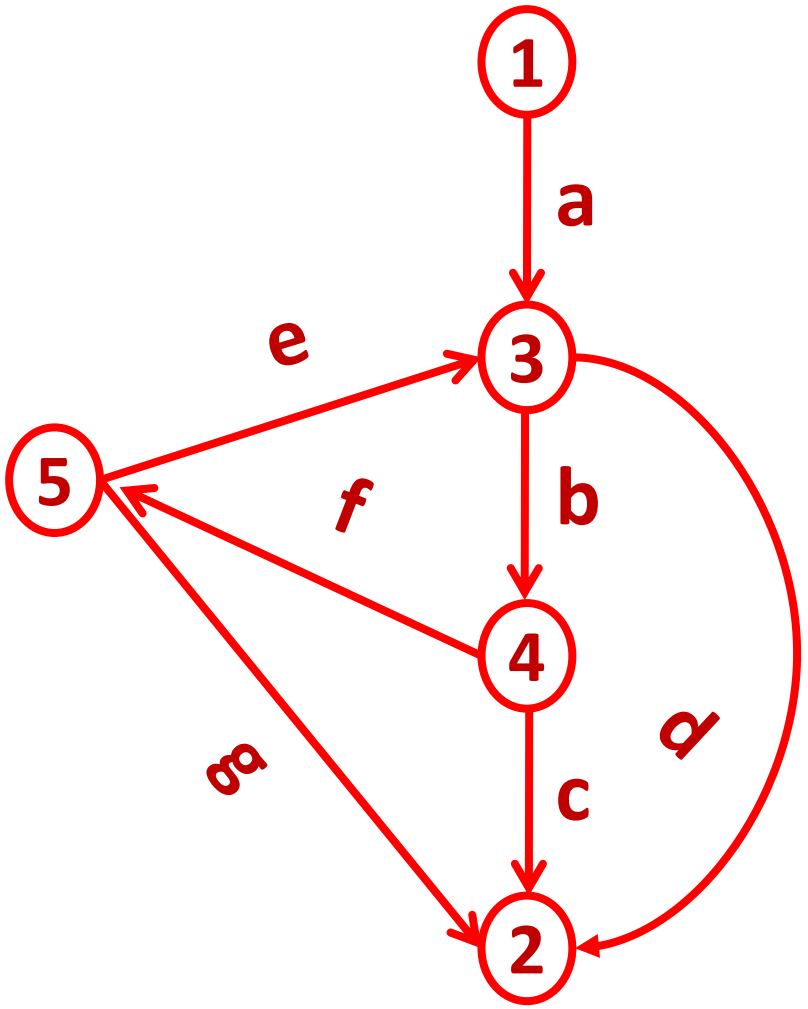
Derive Test Cases …

- We create flow graph using symbols and construction rules.
- It is created by numbering procedural design language statement that maps into nodes.
- Cyclomatic complexity of flow graph is determined.
- These number provides us the number of independent path to control structure.
- After this we design test cases such as condition at predicate nodes are tested.

Derive Test Cases …

- Graph Metrics is a square matrix in which the number of rows and columns is equal to number of nodes in flow graph.
- Each row and column corresponds to an identified node and a matrix entry corresponds to a connection between nodes.
- Consider a flow graph and graph matrix which is given below.
- Each node in the flow graph is identified by number and each edge is identified by alphabets.
- We can add link weight to graph matrix to provide an additional information of control flow.

Graph Metrics



Nodes — Connected to

	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		g	e		

Controlled Structured Testing

- It broadens testing coverage and improves quality of “White Box Testing”.
- Various “**Controlled Structured Testing**” are: -
  - Basis Path Testing
  - Conditional Testing
  - Data Flow Testing
  - Loop Testing

Basis Path Testing

- Already Discussed

Conditional Testing

- It is a test case design method that exercises the logical conditions contained in a program module.
- A simple condition is a boolean variable or a relational expression, generally processed with boolean operator like NOT, OR or AND.
- A relational expression takes the form E_1< relational operator> E_2, where E_1 and E_2 are arithmetic expressions and relational operator is any one of the <, ≤,=,>, ≥, ≠.

Conditional Testing …

- A compound condition is composed of two or more simple condition, boolean operators or parenthesis.
- We assume that Boolean operators allowed in compound conditions include OR(|), AND(&), NOT(¬).
- In conditional testing, following testing strategies are usually used:-
  - Branch Testing
  - Domain Testing
  - BRO Testing

Conditional Testing: Branch Testing

- It tests true and false branches of any condition C.

Conditional Testing: Domain Testing

- It tests relational boolean expression with multiple (say 3 – 4) test values.

Conditional Testing: BRO Testing

- It tests Branch and Relation for any error by using conditional constraints for a condition C.

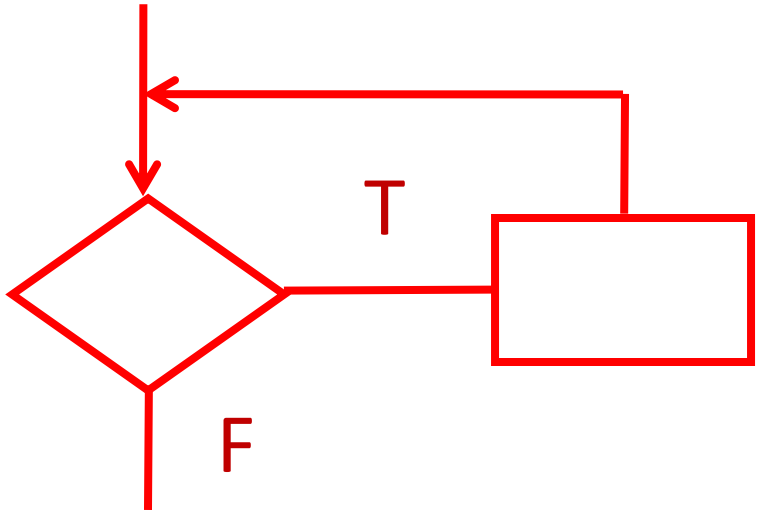
Loop Testing

- It is a white box testing that focuses on the validation of loop construct.
- There are four categories of loops. They are: -
  - Simple Loop
  - Concatenated Loop
  - Nested Loop
  - Unstructured Loop

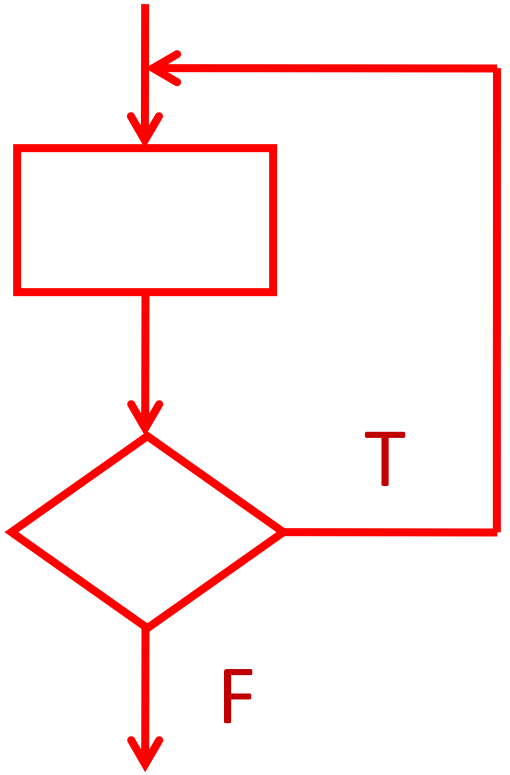
Simple Loop Testing

- The following sets of test case can be applied to simple loop where n= maximum number of allocated passes through the loop.
  - Skip the loop entirely
  - Only one pass through the loop
  - Two passes through the loop
  - M-passes through the loop
  - (n-1),n,(n+1) passes through the loop

Simple Loop Testing …



Entry Controlled



Exit Controlled

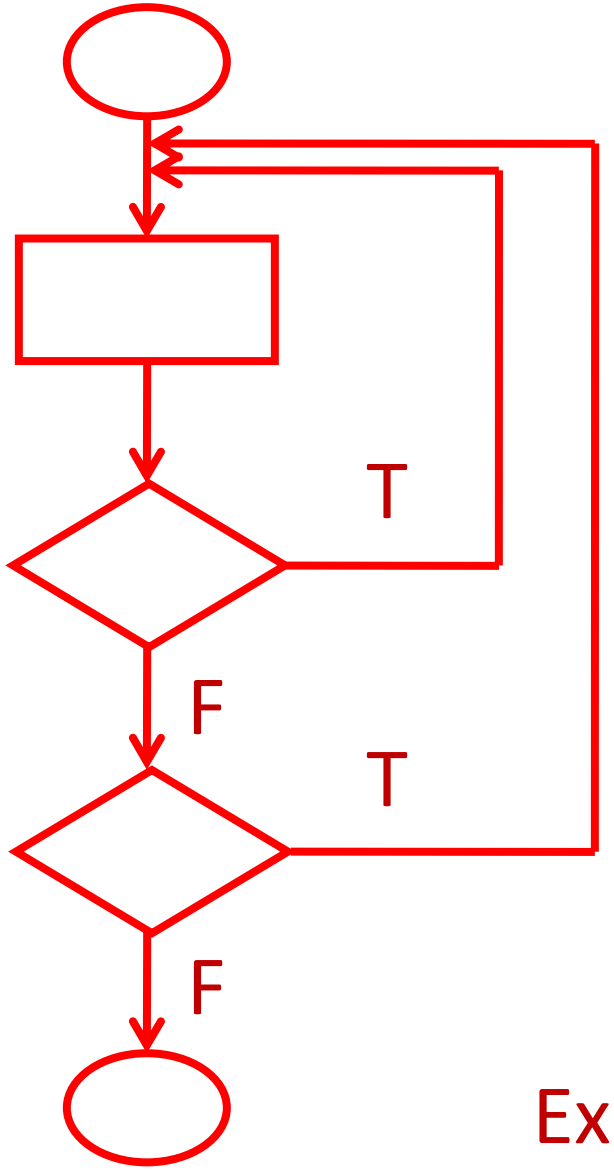
Nested Loop Testing

- The number of possible tests would geometrically grow as level of testing increases. This would result in an impractical number of test.
- Beizer suggested an approach which would help to reduce the number of tests.
  - Start at the inner most loop, set all the other loops to minimum value.
  - Conduct simple loop test for the inner most loop which holding the outer loop at their minimum iteration parameter (loop counter) values.

Nested Loop Testing …

- Another test for out of range or excluded values.
- Work outward conducting test for the next loop but keeping all the other outer loops at minimum value and other nested value to typical values.
- Continue until all the loop have been tested.

Nested Loop Testing …

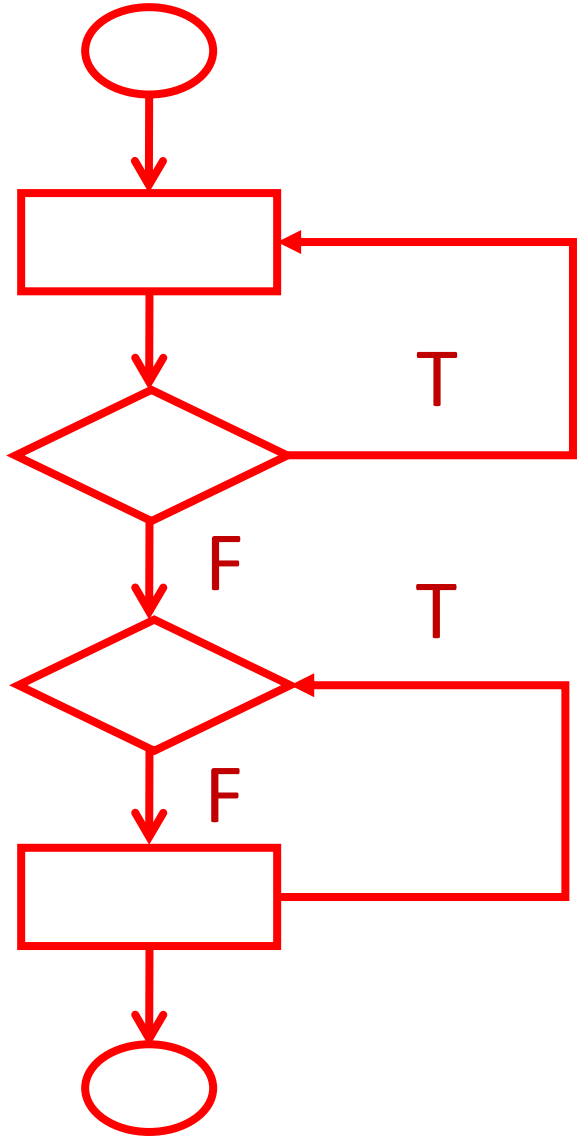


Exit Controlled

Concatenated Loop Testing

- It can be tested using the approach defined by simple loop.
- If each of the loops is independent of each other, however, if two loop are concatenated and loop counter for loop 1 is used as a initial value for loop 2 then the loop are not independent. Then nested loop testing is recommended.

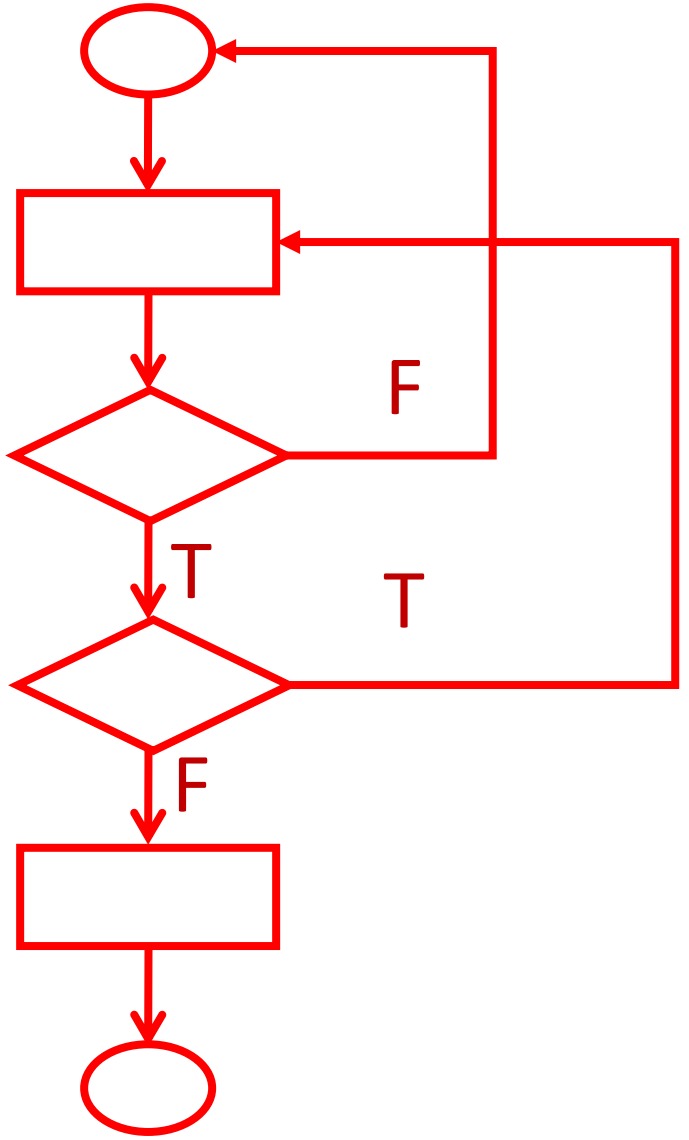
Concatenated Loop Testing …



Unstructured Loop Testing

- We usually redesign such class of loops to reflect the use of structured programming construct.

Unstructured Loop Testing …



Data Flow Testing

- A data flow testing method select test path of a program according to locating of definition and uses a variable in the program.
- To illustrate data flow testing, assume that each statement in a program is assigned a unique statement number and that each function does not modify its parameter or global variable.
  - Data Error Flow(DEF)= { x| statement contains definition of x}
  - USE(s)={x| statement S contains a use of x}

Data Flow Testing …

- Statement S is an if – else loop statement, then its DEF(s) set is an empty set and USE set is based on the condition of statement S.
- The definition of variable x at statement S is said to be “live - at statement S prime(S’)”.
- If there exist a path from statement S to statement S’, that contain no other definition of S’. A definition use chain of variable x which is of the form [x, S, S’], where S and S’ are statement number.

How to Perform Black Box Testing?

- Test case Design
- Black Box test case designs are
  - Graph Based Testing Design
  - Equivalent Partitioning
  - Boundary Value Analysis
  - Comparison Testing

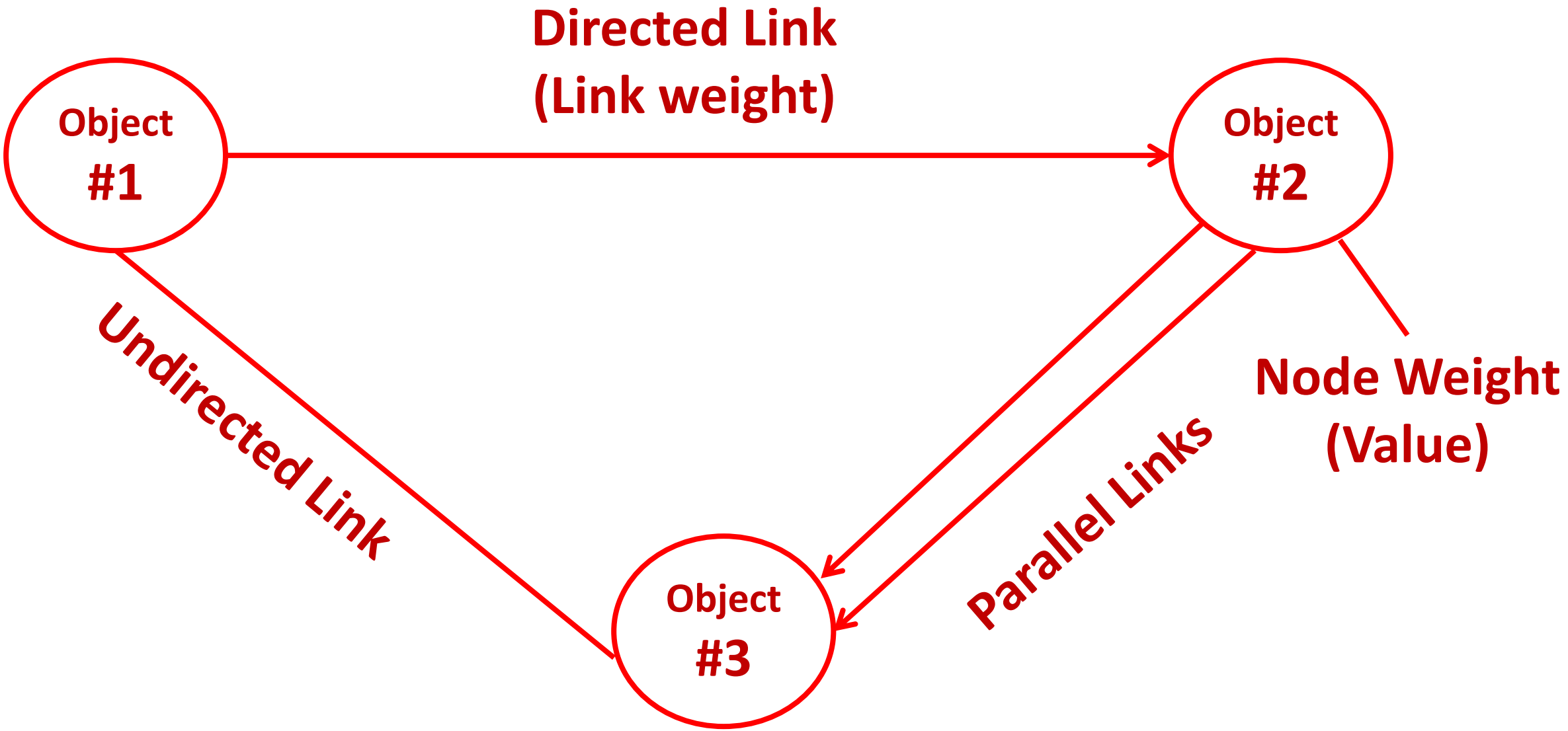
Graph Based Testing Design

- The first attempt in black box testing is to understand the objects that are modeled in the software and the relationships that connects these objects.
- After this, we define a series of test that verify all objects have expected relationship.
- We create a graph by making nodes that represent object, link represent relationship, node weights that describes the properties of the nodes.

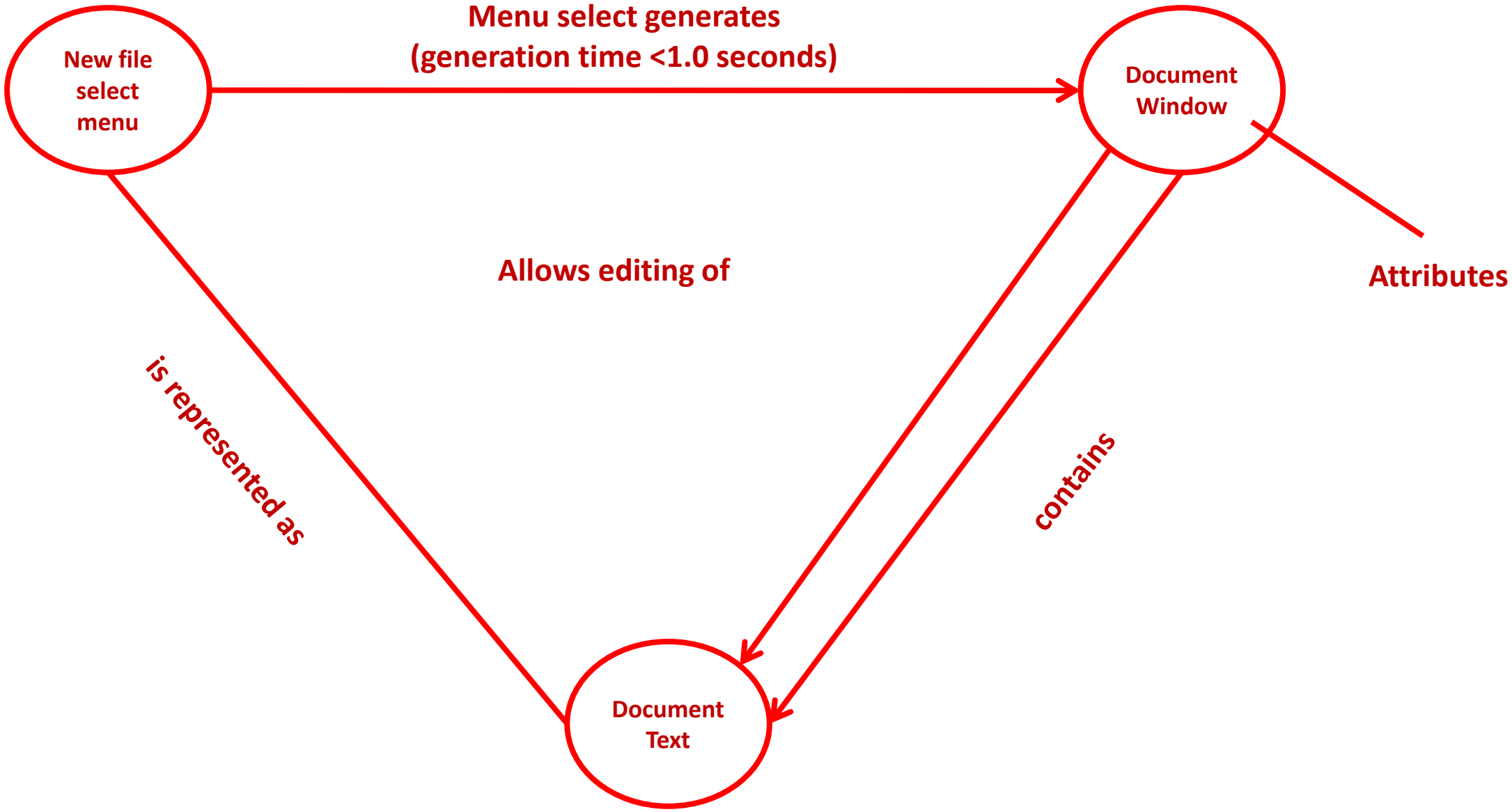
Graph Based Testing Design …

- In the following graph, a collection of nodes represents objects.
- Links that represent relationship between objects.
- Node weights that decides the properties of a node and link weights describe some characteristics of the link.
- A directed link represented with an arrow indicated that the relationship moves only in one direction.

Graph Based Testing Design …



Graph Based Testing Design …



Graph Based Testing Design …

- A menu selects a new file generates a document window.
- The node weight of document window provides a list of window attributes that are expected when the window is generated.
- The link weight indicates that the window must generated in less than 1.0 seconds, an undirected link establishes a symmetric relationship between a new file menu selection and document text, parallel link indicates the relationship between document window and document text.

Equivalent Partitioning

- It is a black box testing method that divides the input domain of a program into classes of data from which the test case(s) can be derived.
- Equivalent partitioning defines a test case that uncovers classes of errors there by reducing total number of test cases.
- Equivalence classes may be defined according to following guidelines
  - If the input condition specifies the range, one valid and two invalid classes are defined.

Equivalent Partitioning …

- If the input condition specifies a value, one valid and one invalid class are defined.
- If the input condition specifies a member of a set, one valid and one invalid classes are defined.
- If the input condition is boolean, one valid and one invalid classes are defined.

Boundary Value Analysis

- Huge number of errors occur at the “boundary of input domain” rather than in the “center”.
- Due to this the Boundary Value Analysis(BVA) has been developed as a testing technique.
- It leads to the selection of test cases that exercise boundary value.
- It is a test case design technique that complements equivalence partitioning.
- In stead of selecting any element of an equivalence class, it leads to the selection of test cases at the edges of the class.

Boundary Value Analysis …

- Focusing solely on input conditions, it derives test cases from the output domain as well.
- It may be defined according to following guidelines
- If an input condition specifies a range bounded by values a and b, test cases should be designed with values a and b as well as just above and just below.
- If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values just above and below minimum and maximum are also tested.

Boundary Value Analysis …

- Apply guidelines 1 and 2 to output condition. For example, let a temperature v/s pressure table is required as output from an engineering analysis program. Test cases should be designed to create an output report that produces that maximum and minimum allowable number of table entries.
- If internal program data structures have prescribed boundaries for example an array has a defined limit of 100 entries. Be certain to design a text case to exercise the data structure at its boundary.

Comparison Testing

- In some situations, software reliability is very critical.
- In such cases, hardware and software are used to minimize errors.
- For such system, we use an independent version of software developed for them.
- Those independent version form basis at black box testing called comparison testing.
- It is called black to black testing.

Comparison Testing …

- In this, we test multiple implementation of some specification by using various black box testing and their output is compared.
- If output is different, each application is investigated for details.

Testing Based on Matrix Testing

- Quantitative measure that helps to estimate the progress, quality and health of software.
- Metric defines in quantitative terms the degree to which a system, system component or process possesses the given attribute(s).
- Improves the efficiency and effectiveness of a software testing process.
- Quantitative indication of extent, capacity, dimension, amount or size of attribute(s) of a process or product.

Role of Matrix Testing

- Helps to take decision for next phase of activities.
- Understands the type of improvement required.
- Take decision on process or technology change.
- Helps to predict.

Types of Matrix

- Process Metrics
- Product Metrics
- Project Metrics

Examples of Testing Matrix

- Test case execution productivity metrics
- Test case preparation productivity metrics
- Defect metrics
- Defects priority metrics
- Defect by severity metrics

What is Black Box Testing?

- The method of software testing that examines the functionality of an application based on the specifications.
- Also termed as Specifications based testing.
- In Black box testing, the functionality of software is examined without peeping into its internal structure or coding.
- The Software Requirement Specification document acts as the primary source of black box testing.

Who Performs Black Box Testing?

- Independent Testing Team performs Black Box Testing.

When Black Box Testing is done?

- Black Box Testing is usually undertaken during the software testing life cycle.
- The testing can be applied to each and every level of software testing such as unit, integration, system and acceptance testing.

How Black Box Testing is done?

- In this method, testing team selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not.
- In case of the function produces correct output, then it is passed in testing, otherwise failed.
- The testing team reports the result to the development team and then tests the next function.
- After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

Methods of Black Box i.e. Behavioral Testing

- Decision Tables
- Boundary Value Analysis
- State Transition Technique
- All – Pair Testing
- Cause – Effect Testing Technique
- Equivalence Partitioning Technique
- Error – Guessing Technique
- Use Case Technique

Methods of Black Box i.e. Behavioral Testing

- Domain Tests
- Orthogonal Arrays
- Exploratory Testing

Methods of Black Box i.e. Behavioral Testing …

Method	Working
• Decision Table Technique	• Systematic approach where various input combinations and their respective system behavior are captured in a tabular form. • It is appropriate for the functions that have a logical relationship between two and more than two inputs.
• Boundary Value Technique	• Used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. • It tests, while entering boundary value whether the software is producing correct output or not.
• State Transition Technique	• Used to capture the behavior of the software application when different input values are given to the same function. • The method is applied to the applications that provide the specific number of attempts to access the application.

Methods of Black Box i.e. Behavioral Testing …

Method	Working
• All – Pair Testing Technique	• Used to test all the possible discrete combinations of values. • This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
• Cause – Effect Technique	• Underlines the relationship between a given result and all the factors affecting the result. • It is based on a collection of requirements.
• Equivalence Partitioning Technique	• Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
• Error Guessing Technique	• Error guessing is a technique in which there is no specific method for identifying the error. • It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.

Methods of Black Box i.e. Behavioral Testing …

Method	Working
• Use Case Technique	• To identify the test cases from the beginning to the end of the system as per the usage of the system. • The test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end.
• Domain Testing	• Domain is a set of possible values of an independent variable or the variables of a function. • Programs as input data classifiers: domain testing attempts to determine whether the classification is or is not correct. • Domain testing can be based on specifications or equivalent implementation information
• Orthogonal Array Testing	• The purpose of this testing is to cover maximum code with minimum test cases. • Test cases are designed in a way that can cover maximum code as well as GUI functions with a smaller number of test cases.

Methods of Black Box i.e. Behavioral Testing …

Method	Working
• Exploratory Testing	• Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, test design and test execution. • The style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project.