	(ii)	**Explain decision table with suitable example.**	**4M**
	Ans:	A decision table is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a cause-effect‘ table. The first task is to identify a suitable function or subsystem which reacts according to a combination of inputs or events. The system should not contain too many inputs otherwise the number of combinations will become unmanageable. It is better to deal with large numbers of conditions by dividing them into subsets and dealing with the subsets one at a time. Once you have identified the aspects that need to be combined, then you put them into a table listing all the combinations of True and False for each of the aspects. Following decision table shows the sample example of a credit card black box testing	**(Decision table: description: 2marks, example: 2 marks)**

Condition	**Rule1**	**Rule2**	**Rule 3**	**Rule 4**
Pin Number	T	T	T	F
Payment Detail	T	F	F	T
Overdue details	F	T	T	F

OR

i). Decision table testing is black box test design technique to determine the test scenarios for complex business logic.

ii). Decision tables provide a systematic way of stating complex business rules, which is useful for developers as well as for testers.

iii). Decision tables can be used in test design whether or not they are used in specifications, as they help testers explore the effects of combinations of different inputs and other software states that must correctly implement business rules.

iv). It helps the developers to do a better job can also lead to better relationships with them.

v). Testing combinations can be a challenge, as the number of combinations can often be huge.

vi). Testing all combinations may be impractical if not impossible.

vii). We have to be satisfied with testing just a small subset of combinations but making the choice of which combinations to test and which to leave out is also important.

viii). If you do not have a systematic way of selecting combinations, an arbitrary subset will be used and this may well result in an ineffective test effort.

Example of decision table :

In each column of two conditions mention –Yes‖ or –No‖, user will get here four combinations (two to the power of the number of things to be combined). Because of this, it‘s always good to take small sets of combinations at once. To keep track on combinations, give alternate –Yes‖ and –No‖ on the bottom row, put two –Yes‖ and then two –No‖ on the row above the bottom row, etc., so the top row will have all –Yes‖ and then all –No‖ (Apply the same principle to all such tables).

ii) Automated testing can simulate tens, hundreds or thousands of virtual users interacting with network or web software and applications.

Disadvantages of using automated tools for testing are as follows:

1. High investments required in package purchasing and training.
2. High package development investment costs.
3. High manpower requirements for test preparation.
4. Considerable testing areas left uncovered.

c) **What is load testing and stress testing? Describe with respect to system testing.** **4M**

Ans: **Stress testing** is testing the software under less than ideal conditions. So subject your software to low memory, low disk space, slow cpus, and slow modems and so on. Look at your software and determine what external resources and dependencies it has. Stress testing is simply limiting them to bare minimum. With stress testing you starve the software. **(load testing : 2 marks, stress testing : 2 marks)**

For e.g. Word processor software running on your computer with all available memory and disk space, it works fine. But if the system runs low on resources you had a greater potential to expect a bug. Setting the values to zero or near zero will make the software execute different path as it attempt to handle the tight constraint. Ideally the software would run without crashing or losing data.

Load testing is testing the software under customer expected load. In order to perform load testing on the software you feed it all that it can handle. Operate the software with largest possible data files. If the software operates on peripherals such as printer, or communication ports, connect as many as you can. If you are testing an Internet server that can handle thousands of simultaneous connections, do it. With most software it is important for it to run over long periods. Some software should be able to run forever without being restarted. So Time acts as a important variable.

Stress testing and load testing can be best applied with the help of automation tools**.**

Stress testing and load testing are the types of performance testing.

The Microsoft stress utility program allows you to individually set the amounts of memory, disk space, files and other resources available to the software running on the machine.

Example: Open many number of browsers in the windows simultaneously.
Connect more than the specifies clients to the server.
Connect more than one printer to the system.

d) **List what are the different guidelines to be followed while selecting dynamic test tools.** **4M**

Ans: i) Assessment of the organization's maturity (e.g. readiness for change); **(Guidelines: 4marks)**
ii) Identification of the areas within the organization where tool support will help to improve testing processes;
iii) Evaluation of tools against clear requirements and objective criteria;
iv) Proof-of-concept to see whether the product works as desired and meets the requirements and objectives defined for it;
v) Evaluation of the vendor (training, support and other commercial aspects) or open-source network of support;
vi) Identifying and planning internal implementation (including coaching and mentoring for those new to the use of the tool).

3. Test coding standards:
   a) Enforce right type of initialization
   b) Stipulate ways of naming variables.
   c) Encourage reusability of test artifacts
   d) Provide standard interfaces to external entities like operating system, hardware and so on.
4. Test reporting standard: All the stakeholders must get a consistent and timely view of the progress of tests. It provides guidelines on the level of details that should be present in the test report, their standard formats and contents.

b. **Attempt any ONE of the following:** **6**

(i) **Explain V model with diagram.** **6M**

Ans: V model means verification and validation model. It is sequential path of execution of processes. Each phase must be completed before the next phase begins.

Under V-model, the corresponding testing phase of the development phase is planned in parallel. So there is verification on one side of V & validation phase on the other side of V.

(Diagram : 2 marks, verification Explanation: 2 marks, validation Explanation: 2 marks)

Verification Phase:

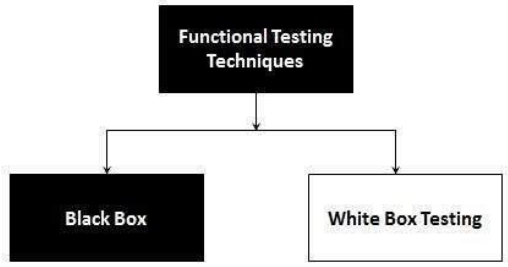
1. **Overall Business Requirement:** In this first phase of the development cycle, the product requirements are understood from customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirements. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
2. **Software Requirement:** Once the product requirements are clearly known, the system can be designed. The system design comprises of understanding & detailing the complete hardware, software & communication set up for the product under development. System test plan is designed based on system design. Doing this at earlier stage leaves more time for actual test execution later.
3. **High level design:** High level specification are understood & designed in this phase. Usually more than one technical approach is proposed & based on the technical & financial feasibility, the final decision is taken. System design is broken down further into modules taking up different functionality.
4. **Low level design:** In this phase the detailed integral design for all the system modules is specified. It is important that the design is compatible with the other modules in the system & other external system. Components tests can be designed at this stage based on the internal module design,
5. **Coding:** The actual coding of the system modules designed in the design phase is taken up in the coding phase. The base suitable programming language is decided base on requirements. Coding is done based on the coding guidelines & standards.

Validation:

1. **Unit Testing:** Unit testing designed in coding are executed on the code during this validation phase. This helps to eliminate bugs at an early stage.
2. **Components testing:** This is associated with module design helps to eliminate defects in individual modules.
3. **Integration Testing:** It is associated with high level design phase & it is performed to test the coexistence & communication of the internal modules within the system

3. **Functional testing**: Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

There are two major Functional Testing techniques as shown below:



4. **Compatibility testing**: Compatibility testing is a non-functional testing conducted on the application to evaluate the application's compatibility within different environments. It can be of two types - forward compatibility testing and backward compatibility testing.
   - Operating system Compatibility Testing - Linux , Mac OS, Windows
   - Database Compatibility Testing - Oracle SQL Server
   - Browser Compatibility Testing - IE , Chrome, Firefox
   - Other System Software - Web server, networking/ messaging tool, etc.

6. **Attempt any FOUR of the following:** **16**

a) **List the different techniques to detect defects. Describe any two of them.** **4M**

Ans: **(List: 1 mark, Describing: 1.5 marks each)**

Static Techniques: Static techniques of quality control define checking the software product and related artifacts without executing them. It is also termed _desk checking/verification /white box testing‘. It may include reviews, walkthroughs, inspection, and audits Here; the work product is reviewed by the reviewer with the help of a checklist, standards, any other artifact, knowledge and experience, in order to locate the defect with respect to the established criteria. Static technique is so named because it involves no execution of code, product, documentation, etc. This technique helps in establishing _conformance to requirements _view.
Dynamic Testing: Dynamic testing is a validation technique which includes dummy or actual execution of work products to evaluate it with expected behavior. It includes black box testing methodology such as system testing and unit testing. The testing methods evaluate the product with respect to requirements defined, designs created and mark it as _pass _or _fail‘. This technique establishes _fitness for use‘ view.
Operational techniques: Operational techniques typically include auditing work products and projects to understand whether the processes defined for development /testing are being followed correctly o not, and also whether they are effective or not. It also includes revisiting the defects before and after fixing and analysis. Operational technique may include smoke testing and sanity testing of a work product.

- Programmers love code coverage. It allows them to attach a number—an actual, hard, real number, such as ***75%***—to the performance of their unit tests, and they can challenge themselves to improve the score.
- Meanwhile, looking at the code that ***isn't*** covered also can yield opportunities for improvement and bugs!

ii. Weaknesses

- Customer-level coverage tools are expensive, programmer-level tools that tend to assume the team is doing automated unit testing and has a continuous-integration server and a fair bit of discipline.
- After installing the tool, most people tend to focus on statement coverage—the least powerful of the measures.
- Even decision coverage doesn't deal with situations where the decision contains defects, or when there are other, hidden equivalence classes; say, in the third-party library that isn't measured in the same way as your compiled source code is.
- Having code-coverage numbers can be helpful, but using them as a form of process control can actually encourage wrong behaviours. In my experience, it's often best to leave these measures to the programmers, to measure optionally for personal improvement (and to find dead spots), not as a proxy for actual quality.

g) Regression and High-Volume Test Techniques

- People spend a lot of money on regression testing, taking the old test ideas described above and rerunning them over and over.
- This is generally done with either expensive users or ***very*** expensive programmers spending a lot of time writing and later maintaining those automated tests.

i. Strengths

- For the right kind of problem, say an IT shop processing files through a database, this kind of technique can be extremely powerful.
- Likewise, if the software deliverable is a report written in SQL, you can hand the problem to other people in plain English, have them write their own SQL statements, and compare the results.
- Unlike state-transition diagrams, this method shines at finding the hidden state in devices. For a pacemaker or a missile-launch device, finding those issues can be pretty important.

ii. Weaknesses

- Building a record/playback/capture rig for a GUI can be extremely expensive, and it might be difficult to tell whether the application hasn't broken, but has changed in a minor way.
- For the most part, these techniques seem to have found a niche in IT/database work, at large companies like Microsoft and AT&T, which can have programming testers doing this work in addition to traditional testing, or finding large errors such as crashes without having to understand the details of the business logic.
- While some software projects seem ready-made for this approach, others...aren't.
- You could waste a fair bit of money and time trying to figure out where your project falls.