Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q.1.

A. Attempt any Three

a) Define "software testing". Why is it required?(Defination-1 Mark, Need- 3 Marks)

Software Testing: Testing can be defined in simple words as "Performing Verification and Validation of the Software Product" for its correctness and accuracy of working.

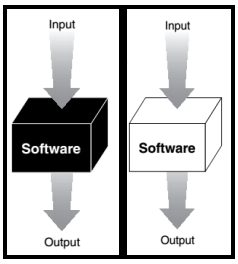
1. The testing of software is an important means of assessing the software to determine its quality.
2. Since testing typically consumes 40% to50% of development efforts, and consumes more effort for systems that require higher levels of reliability, it is a significant part of the software engineering.
3. With the development of fourth generation languages (4GL), which speeds up the implementation process, the proportion of time devoted to testing increased.
4. As the amount of maintenance and upgrade of existing systems grow, significant amount of testing will also be needed to verify systems after changes are made.

b) Differentiate between Black Box and white Box testing.

(Students may write the answer in tabular format)(Any four points 1 M each)
(Black Box Testing – 2 Marks, White Box Testing – 2 Marks)

Black Box Testing

1. This approach tests all possible combinations of end-user actions.
2. Black box testing assumes no knowledge of code and is intended to simulate the end-user experience.
3. The tester can use sample applications to integrate and test the application block for black box testing.
4. Planning for black box testing immediately after the requirements and the functional specifications are available.
5. Black box testing should be conducted in a test environment close to the target environment.



Black box testing White Box Testing

White box testing

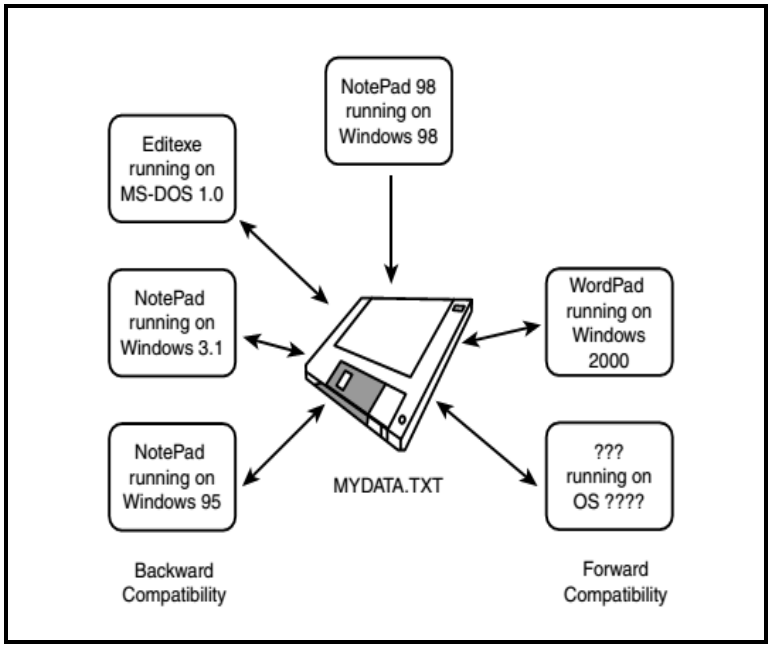
1. This is also known as glass box, clear box, and open box testing.
2. In white box testing, test cases are created by looking at the code to detect any potential failure scenarios.
3. The suitable input data for testing various APIs and the special code paths that need to be tested by analysing the source code for the application block.
4. Therefore, the test plans need to be updated before starting white box testing and only after a stable build of the code is available.
5. White box testing assumes that the tester can take a look at the code for the application block and create test cases that look for any potential failure scenarios.
6. During white box testing, analyze the code of the application block and prepare test cases for testing the functionality to ensure that the class is behaving in accordance with the specifications and testing for robustness.
7. A failure of a white box test may result in a change that requires all black box testing to be repeated and white box testing paths to be reviewed and possibly changed.

c) With an example explain how will you test formula and equation?
(2 marks – Formula, 2 Marks- Equation)
Note: This is an example from the examiners point of view; any relevant answer should be given marks
Ans: To test the formula and the equation are defined as a part of boundary and sub boundary conditions. They can be checked for computational errors.
For example

1. The formula $b^2 – 4ac$ can be tested for the various values of a, b, c
2. The output of this formula can be given as one of the values to the equation
   $-b + b^2 - 4ac/2a$ or $–b – b^2 – 4ac/2a$
3. The values of the formula are the sub boundary conditions on which the formula is tested.
4. The larger scope lies with the equation as it tests for boundary conditions.

d) Define terms "Backward" and "Forward" compatibility.
(Backward Compatibility – 2 marks, Forward Compatibility- 2 Marks)

1. In backward compatibility, the software works with previous versions of the software.
2. In forward compatible, the software works with future versions of the software.
3. The simplest demonstration of backward and forward compatibility is with a .txt or text file.
4. As shown in Figure, a text file created using Notepad 98 running under Windows 98 is backward compatible all the way back to MS-DOS 1.0.
5. It's also forward compatible to Windows 2000 and its future version.



B. Attempt any one.

a) What is meaning of "test to pass" and "test to fail"? Give an example.
(Test to pass – 3 Marks, Test to fail – 3 Marks)
(Any other suitable example may also be considered)

There are two fundamental approaches to testing software test-to-pass and test-to-fail.

1. When you test-to-pass, you really assure only that the software minimally works and as a software tester it is the first positive approach.
2. Don't push its capabilities; don't see what you can do to break it. Test the software by applying the simplest and most straightforward test cases.
3. The software should be tested to work under the best ideal condition.
4. During test to fail, the software should be tested for the boundary conditions on its various limits.
5. If the program is using integer numbers, than the output should be tested in the limits of the integer variable.

For example

```
1. #include<stdio.h>
2. void main()
3. {
4. int i , fact= 1, n;
5. printf("enter the number ");
```

```
6.  scanf("%d",&n);
7.  for(i =1 ;i <=n;i++)
8.  fact = fact * i;
9.  printf ("the factorial of a number is ➔"%d", fact);
10. }
```

The above program works for the integer values from 1 to 7, and fails as soon as the input value is entered as 8, as it exceeds the integer range of variable for fact.

b) What is meaning of a Walkthrough? How is it carried out? What are its advantages? (Four points – 4 marks)

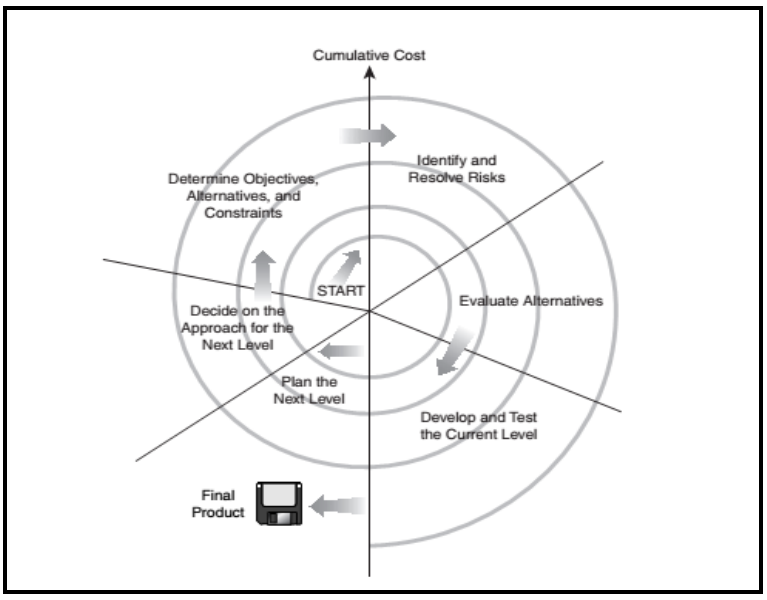
1. Walkthroughs are the next step up in formality from peer reviews.
2. In a walkthrough, the programmer who wrote the code formally presents (walks through) it to a small group of five or so other programmers and testers.
3. The reviewers should receive copies of the software in advance of the review so they can examine it and write comments and questions that they want to ask at the review.
4. Having at least one senior programmer as a reviewer is very important.
5. The advantage of the walkthrough is it gives a rough layout of the final demonstration and helps find if any unsaid and not reviewed errors.

Q.2. Attempt any four.

a) With neat sketch, explain spiral model of software development. (Explanation – 2 Marks, Diagram – 2 Marks)

1. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.
2. A software project repeatedly passes through these phases in iterations.
3. The baseline spiral starting in the planning phase, requirements are gathered and risk is assessed.
4. The general idea behind the spiral model is that you don't define everything in detail at the very beginning.
5. Start small, define your important features, try them out, get feedback from your customers, and then move on to the next level.
6. Repeat this until you have your final product.

Diagram of Spiral Model



b) How can research on standards and guidelines help testing? Give an example. Site where such guidelines are available.
Research Existing Standards and Guidelines
(Explanation 2 M, Example 2 M)

1. Research to be done on the existing similar type of software's to gain Knowledge about the various factors that affect its working.
2. Effort to standardize the hardware and the software is important.
3. The result is that we now have products reasonably similar in their look and feel that have been designed.
4. The guidelines are followed for the existing software's are implemented.
   Example site: www.softwaretesting.com, any other relevant site.

c) List computation error. Explain any two with examples.
(Listing- 3 Marks, example- 1 Mark)
Computational or calculation errors are essentially bad math. The calculations don't result in the expected result.

1. Do any calculations that use variables have different data types, such as adding an integer to a floating-point number?
2. Do any calculations that use variables have the same data type but are different lengths—adding a byte to a word, for example?
3. Are the compiler's conversion rules for variables of inconsistent type or length understood and taken into account in any calculations?
4. Is the target variable of an assignment smaller than the right-hand expression?
5. Is overflow or underflow in the middle of a numeric calculation possible?
6. Is it ever possible for a divisor/modulus to be zero?
7. For cases of integer arithmetic, does the code handle that some calculations, particularly division, will result in loss of precision?
8. Can a variable's value go outside its meaningful range? For example, could the result of a probability be less than 0% or greater than 100%?

d) What is configuration testing? How is it done?
(Explain – 4 Marks,)
Configuration testing is the process of various types of hardware. It involves testing the various possible configurations for a standard PC used for personnel and commercial use. The hardware which needs to be tested is listed below.

1. **The PC: -** There are dozens of well-known computer manufacturers, such as Compaq, Dell, Gateway, Hewlett Packard, IBM, and others. Each one designs and builds PCs using their own designed components or parts from other manufacturers.
2. **Components: -**Most PCs are modular and built up from various system boards, component cards, and other internal devices such as disk drives, CD-ROM drives, video, sound, modem, and network cards. There are TV cards and specialized cards for video capture and home automation. There are even input/output cards that can give a PC the ability to control a small factory! These internal devices are built by hundreds of different manufacturers.
3. **Peripherals: -**Peripherals, shown in Figure.2, are the printers, scanners, mouse, keyboards, monitors, cameras, joysticks, and other devices that plug into system and operate externally to the PC.

4. **Interfaces.** The components and peripherals plug into your PC through various types of interface connectors. These interfaces can be internal or external to the PC. Typical names for them are ISA, PCI, USB, PS/2, RS/232, RJ-11, RJ-45, and Firewire. There are so many different possibilities that hardware manufacturers will often create the same peripheral with different interfaces. It's possible to buy the exact same mouse in three different configurations!
5. **Options and memory.** Many components and peripherals can be purchased with different hardware options and memory sizes. Printers can be upgraded to support extra fonts or accept more memory to speed up printing. Graphics cards with more memory can support additional colors and higher resolutions. The system board can have different versions of BIOS (its Basic Input Output System) and, of course, various amounts of memory.
6. **Device Drivers**. All components and peripherals communicate with the operating system and the software applications through low-level software called device drivers. These drivers are often provided by the hardware device manufacturer and are installed when you set up the hardware. Although technically they are software, for testing purposes they are considered part of the hardware configuration.

e) What is a test plan? What is it's need?
(Any four points – 1 Mark each)

1. Performing testing tasks would be very difficult if the programmers wrote their code without telling what it does, how it works, or when it will be complete.
2. Software testers don't communicate what tester plans to test, what resources you need, and what your schedule is, your project will have little chance of succeeding.
3. The software test plan is the primary means by which software testers communicate to the product development team what they intend to do.
4. The test plan is simply a by-product of the detailed planning process that's undertaken to create it. It's the planning process that matters, not the resulting document.
5. The ultimate goal of the test planning process is communicating (not recording) the software test team's intent, its expectations, and its understanding of the testing that's to be performed.

f) Is testing related to organizational structures? If yes, how?
(This is a generalized question, any relevant answer with the justification can be considered)
(Explanation 4M)

1. Testing is related to the organizational structure.
2. As the software developed is with the guidelines and requirements of the organization.
3. The peer review, walkthrough, the dumb and the gorilla and the smart monkey testing strategies are all part of the organizational structure.
4. The test to pass and test to fail conditions are designed or prepared for the requirement given by the organization

Q.3. Attempt any four.

a) Explain with one example, boundary condition testing.
(2 Marks for explanation, 2 Mark for example)

Boundary conditions: Many systems have tendency to fail on boundary. So it is important to test boundary conditions of application.

E.g. If we want to display numbers from 10 to 49, the condition will be, If (n>=10 AND n<50). But if you have given the condition as: if(n>10 AND n<50) then 10 will

not get printed. Or if (n>=10 AND n<=50) then 50 will also get printed. So for this example we have to test 9, 10, 11 and 48,49,50 values of 'n'. These values are respectively lower_boundary-1, lower_boundary, lower_boundary+1, upper_boundary-1, upper_ boundary, upper_boundary+1.

Types of boundary conditions: Numeric, character, position, quality, speed, location, size. For this type the characteristics can be: First/Last, Min/Max, Over/Under, Highest/Lowest.
For each boundary condition include boundary value in at least one valid test case. Include value just beyond boundary in at least one invalid test case. Include test cases with input such that outputs at boundaries are produced.

TESTING THE BOUNDARY EDGES:

- First-1/Last+1
- Start-1/Finish+1
- Largest+1/Smallest-1
- Min-1/Max+1
- Just Over/Just Under
  e.g. If a text entry field allows 1 to 255 characters, try entering 1, 2 character and 254,255 characters as the valid partition. Enter 0 and 256 characters as the invalid partitions.

b) List I/O errors, as a part of code review checklist. .
(1 mark for each error & any four)

Input/output errors:

- These errors include anything related to reading from a file, accepting input from keyboard or mouse & writing to an output device such as printer or screen.
- The items presented here are very simplified & generic. You should adopt & add to them properly.
- Does software strictly adhere to the specified format of data being read or written by external device?
- If the file or peripheral isn't present or ready? Is that error condition handled?
- Does the software handled the situation of the external device being disconnected not available or full during a read or write?
- Are all conceivable errors handled by software in an expected way?
- Have all errors message been checked for correctness, appropriateness, grammar & spelling?

c) List the components you will test a website. Give an example, from real world.
(2 marks for listing components & 2 marks for example)
Note: Any suitable example should be considered.
Internet web pages are documents of text, picture, sounds, video, and hyperlink. Web page features contains list of elements.

- Text of different sizes, fonts, and colors
- Graphics & photos
- Hyper linked text and graphics
- Varying advertisements
- Drop down selection boxes
- Fields in which the users can enter data

Example: Consider Home page of apple's website www.apple.com .
It has all the basic element- text, graphics, hyperlinks to other pages on the site & hyperlinks to other web site.
If you looked at the site map(www.apple.com/sitemap) you could found links to more than 60 different sub sites, each one with several pages.

d) What is the process of bug reporting? Prepare a sample bug reporting format? (Explanation 2 marks & 2 marks for report)

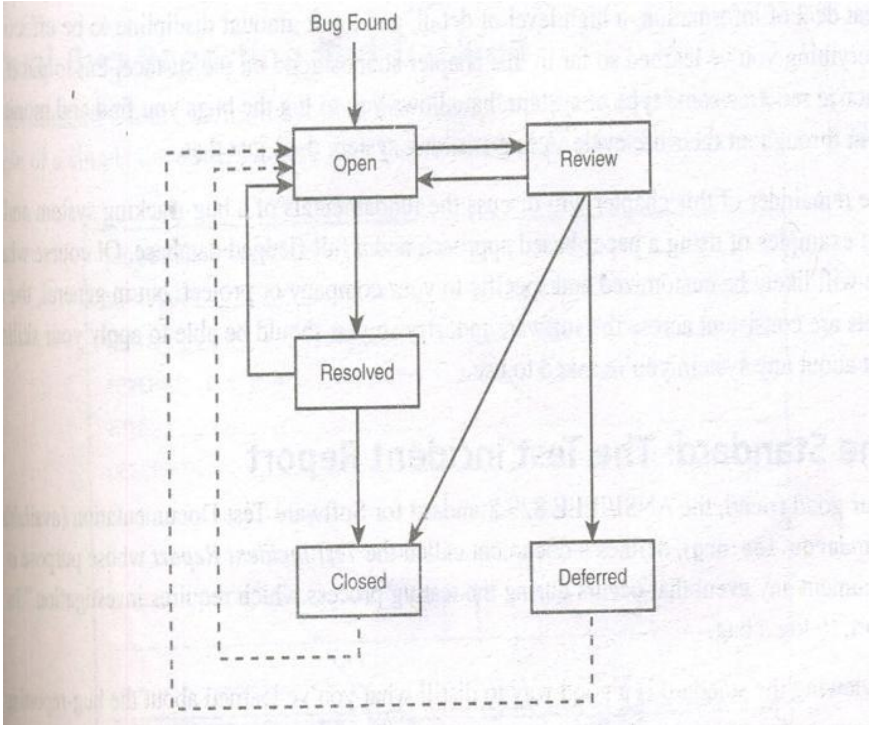
Fig shows what such a paper bug report can look like .
This one page form can hold all information necessary to identify & describe a bug. It also contains fields that you can use to track a bug through its cycle. Once the form is filled by the tester, it can be assigned to a programmer to be fixed. The programmer has fields where she can enter information regarding the fix, including choices for possible resolutions. There is also an area where, once the bug is resolved, the tester can supply information about his efforts in retesting & closing out the bug. At the bottom of the form is an area for signatures. In many industries, you put name on the line to reflect that a bug has been resolved to your satisfaction.

XYZ Software Ltd Bug Report Bug ID_____ SOFTWARE_____
RELEASE_____ ASSIGENED TO_____ Severity:1 2 3 4 Priority: 1 2 3 4 Reproducible: Yes/No Title:_____
Description:_____
_____
_____ Resolution:
Fixed/Duplicate/No-repro/Can't Fix/Deferred/Won't Fix Date Resolved:_____ Resolved by:_____ Version:_____ Resolution Comment:_____
_____
_____ Retested by:_____
Version Tested_____ Date Tested_____

e) Explain bug's life cycle, with neat diagram. (2 marks diagram & 2 marks explanation)

Bug's life Cycle: When a bug is first found by a software tester, it's logged and assigned to a programmer to be fixed. This state is called the open state. Once the programmer fixes the code, he assigns it back to the tester and the bug enters the resolved state. The tester then performs a regression test to confirm that the bug is indeed fixed and, if it is, closes it out. The bug then enters its final state, the closed state.
Review State: It is a state where the project manager or the committee decides whether the bug should be fixed. In some projects all bugs go through the review state before they are assigned to the programmer for fixing. Review state can also go to directly closed state. This happens if the review decides that the bug should not be fixed.
Deferred State: It is the state when the review may determine that the bug should be

considered for fixing at some time in the future but not for this release of the software. The additional line from the resolved state back to the open state covers the situation where the tester finds that the bug has not been fixed. It gets reopened and the bug's life cycle repeats. An error which is closed or deferred may be afterwards a serious bug then it is reopened and starts through the process again.

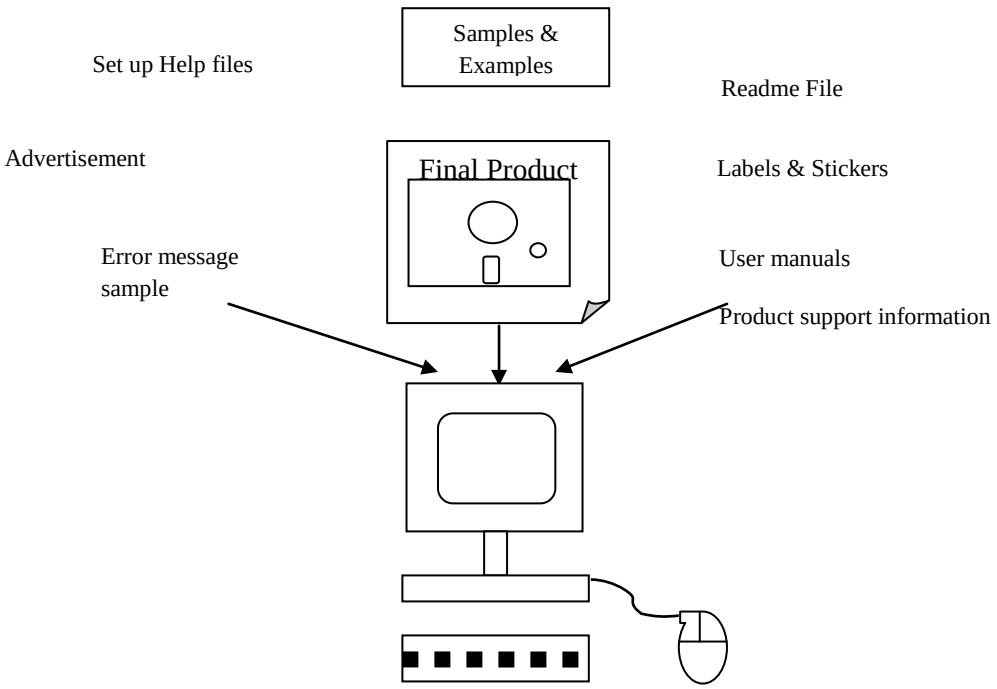


Q.4.

A. Attempt any three.

a) What parts make up a software product?
(Diagram is optional) (1/2 mark each for listing Any 8 listings)

Numerous supporting parts are a shown in fig.



The software floppy disk or CD-ROM is a just one of many pieces that make up software product.
Help Files
Samples & examples
Product support information
Error message setup & installation
User manual
Labels & Stickers
Icon & art
Ads & marketing material
Readme file

b) What is the meaning of equivalence partitioning? Illustrate with an example.
(2 marks for meaning & 2 marks for example)

Equivalence partitioning : Equivalence partitioning is process of methodically reducing the huge set of possible test cases into much smaller but still equally effective. It is also called as equivalence classing for ex. A program which edits credit limits within a given range ($10,000- $15,000) would have three equivalence classes.
Less than $10,000 (invalid)
Between $10,000 & $ 15,000 (valid)
Greater than $15,000 (invalid)

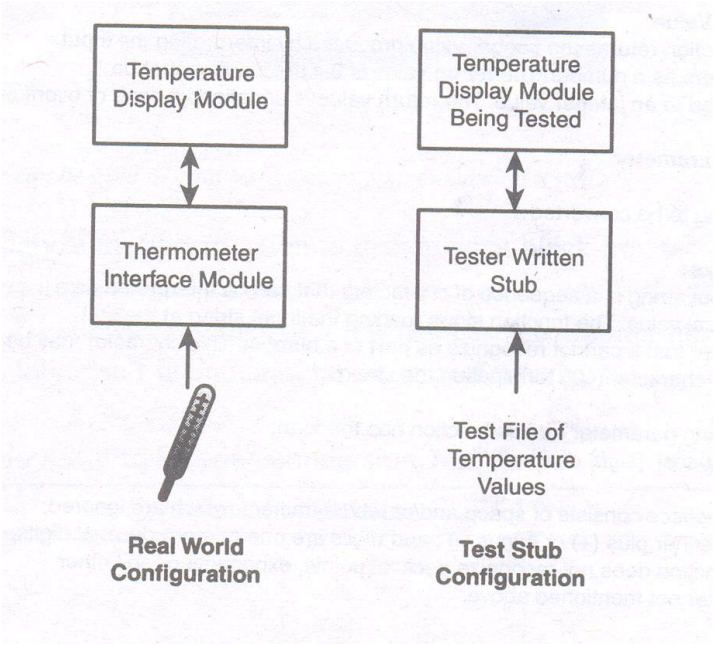
Equivalence classes may be defined according to following guidelines:
1. If an input condition specifies range then one valid and two invalid equivalence classes are defined
2. If an input condition requires specific value the one valid and two invalid equivalence classes are defined.
3. If an input condition is Boolean then one valid and two invalid equivalence classes are defined. The aim of equivalence partitioning is to minimize number of test cases required to cover input condition.

Input condition for above example is 5 digit integer between 10,000 & 99,999 equivalence partitions are <10,000, 10,000-99,999 and >99,999

c) What is integration testing? Give an example.
(2 marks for explanation of integration testing & 2 marks for example)
(Example may vary from student to student)

Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
The purpose of this level of testing is to expose faults in the interaction between integrated units.
Test drivers and test stubs are used to assist in Integration Testing. Integration Testing is performed after Unit Testing and before System Testing. Either Developers themselves or independent Testers perform Integration Testing.
There are mainly two approaches to do integration testing.
i) Top-down Approach
ii) Bottom-up Approach
In this you could write a small piece of code called stubs that acts as interface module

As shown in fig, a low level interface module is used to collect temperature data from an electronic thermometer. Display a module sits right above the interface reads the data from an interface & display it to user. By feeding temperature values from a file directly to display module. With this test stubs configuration, you could run through numerous test values & validate the operation of the display module.



Example2: real time example: in Google
1."Compose"(one module) a mail and send it to some valid user id
2. go check whether the sent mail is there in the "sent items"(second module).
If it's there data flow is correct else data flow is wrong.
In the above example integration between compose and sent items is error free

d) Define "Usability testing". List its advantages.
(Definition 2 marks & advantages 2 marks)
Usability testing is a technique used in interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system. This is in contrast with usability inspection methods where experts use different methods to evaluate a user interface without involving users.
Advantages:
Usability testing focuses on measuring a human-made product's capacity to meet its intended purpose.
Examples of products that commonly benefit from usability testing are foods, consumer products, web sites or web applications, computer interfaces, documents, and devices.
Usability testing measures the usability, or ease of use, of a specific object or set of objects, whereas general human-computer interaction studies attempt to formulate universal principles.

B. Attempt any one.

a) What are the major difficulties faced by a tester, while carrying out configuration testing? What are the solutions?
(This is a generalized question; any relevant answer with the justification should be given marks)
(Explanation 6M)

Configuration testing is the process of checking the operation of software you are testing with all the various types of hardware.

Consider the different configuration possibilities for standard windows based PC used in home & business.

The PC , Components, Peripheral, Interface, Option & Memory, Device Drivers

Major configuration problem can occur for several reason, all requiring someone to carefully examine the code while running the software under different configuration to find bug:

- Your software may have a bug that appears under a broad class of configuration.
- Your software may have bug specific only to one particular configuration.
- The hardware device or its device drivers may have a bug that only your software reveals.
- The hardware device or its device drivers may have a bug that can be seen with lots of other software.

To solve this, following section show the general process that should use when planning your configuration testing.

1. Decide the types of hardware you'll need.
2. Decide what hardware Brands, Models & Device Drivers are available.
3. Decide which hardware features, modes & options are possible.
4. Pare down the identified hardware configuration to a manageable set.
5. Identify your software's unique features that work the hardware configurations.
6. Design the test cases to run on each configuration
7. Execute the tests on each configuration
8. Rerun the tests until the results satisfy your team.

b) List localization issues to be considered in foreign language testing.
(3 marks for each point)

The process of adapting software to a specific locale taking into consideration its language, dialect, local conventions and culture is called localization. Major elements for localization are:

1. Content
2. Data formats.

1. Content: If you are testing a product that will be localized, you need to carefully examine the content to make sure it's appropriate to the area where it will be used. The following list shows various types of contents that you should carefully review for localization issues.
Sample document Icons Pictures Sound Video Help Files Map Marketing materials Packaging web links

2. Data Formats: Different locals use different formats for data units such as currency, time & measurement.

Unit	Consideration
Currency	Different symbols and where they‘re placed
Dates	Order of month, day, year; separators; leading zeros; long and short formats
Times	12-hour or 24-hour, separators

Q.5. Attempt any two.

a) Explain importance of testing by narrating any two software error case studies of your choice.

(**4 marks each for case study.4marks*2 case studies =8marks)**

(NOTE: any of the case study as listed or Y2K, NASA MARS polar lander, intel Pentium floating point division bug will also do.)

Disney's Lion King (1994-1995)

In December 1994, the Disney Company released its first multimedia CD-ROM game for children, the Lion King Animated Storybook. This was Disney's first venture into the market and it was highly promoted and advertised. Sales were huge. It was holiday season for the children. On December 26, the day after Christmas, Disney's customer support phones began to ring, and ring. Soon the phone support technicians were swamped with calls from angry parents with crying children who couldn't get the software to work. Numerous stories appeared in newspapers and on TV news.

It turned out that Disney failed to test the software on a broad representation of the many different PC models available on the market. The software worked on a few systems likely the ones that the Disney programmers used to create the game but not on the most common systems that the general public had.

Patriot Missile Defense System, 1991

The U.S. Patriot missile defense system is a scaled-back version of the Strategic Defense Initiative ("Star Wars") program proposed by President Ronald Reagan. It was first put to use in the Gulf War as a defense for Iraqi Scud missiles. Although there were many news stories touting the success of the system, it did fail to defend against several missiles, including one that killed 28 U.S. soldiers in Dhahran, Saudi Arabia. Analysis found that a software bug was the problem.

A small timing error in the system's clock accumulated to the point that after 14 hours, the tracking system was no longer accurate. In the Dhahran attack, the system had been operating for more than 100 hours.

b) What is a good user interface? Illustrate with an example.

(list traits :2 marks + description of any 4 traits:1.5 marks each= 8 marks)

Good UI is the one which has following 8 traits as explained below:

- Follows standards and guidelines
- Intuitive
- Consistent
- Flexible
- Comfortable
- Correct
- Useful

1. **Follows standards and guidelines**
   The single most important user interface trait is that your software follows existing standards and guidelines or has a really good reason not to. If your software is running on an existing platform such as Mac or Windows, the standards are set. Apple's are defined in the book Macintosh Human Interface Guidelines, published by Addison-Wesley, also available online at developer.apple.com/documentation/mac/HIGuidelines/HIGuidelines-2.html.
2. **Intuitive**
   The computers with the most intuitive UIs are the ones that people don't even realize they're using.
   When you're testing a user interface, consider the following things and how they might apply to gauging how intuitive your software is:

- Is the user interface clean, unobtrusive, not busy? The UI shouldn't get in the way of what you want to do.
- Is the UI organized and laid out well? Does it allow you to easily get from one function to another? Is what to do next obvious? At any point can you decide to do nothing or even back up or back out? Are your inputs acknowledged? Do the menus or windows go too deep?
- Is there excessive functionality? Does the software attempt to do too much, either as a whole or in part? Do too many features complicate your work? Do you feel like you're getting information overload?
- If all else fails, does the help system really help you?

3. **Consistent**
   Consistency within your software and with other software is a key attribute. Users develop habits and expect that if they do something a certain way in one program, another will do the same operation the same way. In Notepad, Find is accessed through the Search menu or by pressing F3. In WordPad, a very similar program, it's accessed through the Edit menu or by pressing Ctrl+F.
4. **Flexible**
   Users like choices not too many, but enough to allow them to select what they want to do and how they want to do it. For example, The Windows Calculator has two views: Standard and Scientific. Users can decide which one they need for their task or the one they're most comfortable using.
5. **Comfortable**

- Appropriateness. Software should look and feel proper for what it's doing and who it's for. A financial business application should probably not go crazy with loud colors and sound effects. A space game, on the other hand, will have much more flexibility with the rules. Software should neither be too bright nor too plain for the task it's intended to perform.
- Error handling. A program should warn users before a critical operation and allow users to restore data lost because of a mistake. People take the Undo/Redo feature for granted today, but it wasn't long ago that these features didn't exist.
- Performance. Being fast isn't always a good thing. More than one program has flashed error messages too quickly to read. If an operation is slow, it should at least give the user feedback on how much longer it will take and show that it's still working and hasn't frozen.

6. **Correct**
   Correctness problems such as this are usually obvious and will be found in your normal course of testing against the product specification. You should pay attention to some areas in particular, however:

- Marketing differences. Are there extra or missing functions, or functions that perform operations different from what the marketing material says? Notice that you're not

comparing the software to the specification you're comparing it to the sales information. They're usually different.

- Language and spelling. Some programmers are poor spellers and writers and often create very interesting user messages. The following is an order confirmation message from a popular e-commerce website hopefully fixed by the time you read this:
  If there are any discrepancies with the information below, please contact us immediately to ensure timely delivery of the products that you ordered.
- Bad media. Media is any supporting icons, images, sounds, or videos that go with your software's UI. Icons should be the same size and have the same palette. Sounds should all be of the same format and sampling rate. The correct ones should be displayed when chosen from the UI.

7. Useful

The final trait of a good user interface is whether it's useful. Remember, here you're not concerned with whether the software itself is useful, just whether the particular feature is. A popular term used in the software industry to describe unnecessary or gratuitous features is dancing bologna

c) List types of Automated test tools. Explain anyone types in detail.
(This is a generalized question. Answers may vary from student to student and should be considered.)
(Listing for 1marks; 7 marks for description)

The list of automated test tools is as given below:

1. Quick test professional.
2. AutoIT
3. Water
4. Selenium
5. Silktest

AutoIT: **AutoIT** is a freeware automation language for Microsoft Windows. In its earliest release, the software was primarily intended to create automation scripts (sometimes called macros) for Microsoft Windows programs but has since grown to include enhancements in both programming language design and overall functionality.
Features:

- Scripting language with BASIC-like structure for Windows Desktop Environment.
- Add-on libraries and modules for specific applications.
- On-line support forum for AutoIT users and developers.
- Supports TCP and UDP protocols.
- Supports COM (component object modeling) objects.
- Call functions in Win32 DLLs.
- Run console applications and access the standard streams.
- Include files in the compiled file to be extracted when run.
- Create GUI interfaces, including message and input boxes.
- Play sounds, pause, resume, stop, seek, get the current position of the sound and get the length of the sound.
- Simulate mouse movements.
- Manipulate windows and processes.
- Automate sending user input and keystrokes to applications, as well as to individual controls within an application.
- Scripts can be compiled into standalone executables.
- Unicode support from version 3.2.4.0.
- 64-bit code support from version 3.2.10.0.

- Supports regular expressions.
- Works with Windows Vista's User Account Control.
- Object oriented design through a library
- Use :
- AutoIt is typically used to produce utility software for Microsoft Windows and to automate routine tasks, such as systems management, monitoring, maintenance, or software installation. It is also used to simulate user interaction, whereby an application is "driven" (via automated form entry, key presses, mouse clicks, and so on) to do things by an AutoIt script.

Q.6. Attempt any four.

a) List white Box and Black Box testing steps in testing library management software in your polytechnic or any other college.
(NOTE :TESTING CASES MAY VARY FROM STUDENT TO STUDENT.)
Black box testing : 2 marks

Black box Testing
Black box testing focuses on the functional requirements of the software. This is black box testing enables the software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program.
Steps for black box testing :

1. Run the software.
2. Click on main menu and click on insert record. Try inserting a book and its record.
3. Click on main menu and click delete a record and check if the record can be deleted.
4. Try updating a record to see for its updation.
5. Exit from software by clicking on Exit.

White Box testing: (2 marks)
White – box testing, sometimes called glass-box testing, is a test case design philosophy that uses the control structure described as part of component-level design to derive test cases. Using White box testing methods it can be ensured that
Steps for white box testing:

1. All independent paths within a module have been exercised at least once
2. Exercise all logical decisions.
3. Execute all loops at their boundaries and within their operational bounds
4. Exercise internal data structures to ensure their validity.

b) Write four important test cases to test login page of your E-mail.
(4 test cases 1 marks each)
(Students may write the answer in tabular format)

Important test cases to test login page of email are as given below:
Test case name: rediffmail login

1. Action : Click on rediffmail link
   Expected result: Username and password page should get displayed
   Actual result: Username and password page is displayed
   Status: pass
2. Action : Click on News link
   Expected result: News page should get displayed
   Actual result: News page is displayed

Status: pass

3. Action : Click on shopping link
   Expected result: Shopping page should get displayed
   Actual result: Shopping page is displayed
   Status : pass
4. Action : Click on sign in link
   Expected result: Sign in dialog box should get displayed
   Actual result: Sign in dialog box is displayed
   Status: pass

c) Write sample test-cases to test "facebook" web site.
NOTE :TESTING CASES MAY VARY FROM STUDENT TO STUDENT.)
(4 sample test cases 4 marks)

Sample test cases for facebook are as given below:

Sr. no	Check item	Test case Objective	Steps to Execute	Test data i/p	Expected Results
1	LOG IN PAGE	Leave all fields as blank as click on login button	Click log in		It should display invalid login or password message on the screen.
2	USERNAME	Enter invalid username	Type in the password and press enter,	Username : JOOHNYY@yahoo.co.in	Invalid username or password should be displayed on the screen
3	Like button	Check the like button	Click on the like button under post	NA	Unlike should appear under the post and your username should be added in the list of users who have liked the post.
4	POST ON WALL	Click the post button when nothing is typed in the post area	Click on Post button	NA	Nothing to display message should be displayed on the screen.
5	Upload a picture	Upload a picture format which is not supported	Add the picture or image to album or on wall	Add the format of the image which is not supported by facebook	Image not supported should be displayed by the screen.

6	Delete a post	Check the delete post activity	Click on the delete post activity	NA	Message asking whether you are sure to delete the post should appear on the screen.
7	Tag the picture	Check the tag activity	Click on the tag button	Na	As you press tag the name of the person should be automatically detected or it should give you the clue of name from your list of friends.

d) Explain stress and load testing with reference to "MSBTE" web site testing.
(stress testing 2 marks and load testing 2 marks)
(NOTE : Generalized question; justified answer should be considered irrespective of detailing.)

Stress testing is the testing where the software under test is starved for the minimal requirements for its basic working.

Stress testing for MSBTE website would include:

1. Running the website on the internet explorer version lesser than that is actually required for its execution.
2. Try downloading the multiple documents with less available memory on hard disk drive.
3. Provide less RAM than that is actually required for the PC on which you have opened the website.
4. Try opening the website on 100s of PCs connected on a single network.
   Load testing is the testing of the software under test by providing it more than the required resources those are required for its normal operation.

Load testing of MSBTE website would include the operations like:

1. Trying to download many more documents at a time on a PC from the website in multiple tabs of a browser.
2. Try connecting multiple printers and print multiple documents from the website.
3. Open the site on multiple PCs connected in a network.
4. Try logging in on the website with the same login and password from different places.

e) Plan your test efforts in the form of a document for "supershop management system" to be tested by you.
(4 points and description 1 marks each)
(Note : Generalized question, any supershop management testing may given marks)
Testing of a supershop management system includes:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

Test Plan

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A test plan will include the following.

- Introduction to the Test Plan document
- Assumptions of data when testing the supershop management system
- List of test cases included in Testing the supershop management system
- List of features to be tested
- What sort of Approach to use when testing the software
- List of Deliverables that need to be tested
- The resources allocated for testing the application
- Any Risks involved during the testing process
- A Schedule of tasks and milestones as testing is started

Test Scenario

A one line statement that tells what area in the application will be tested. Test Scenarios are used to ensure that all process flows are tested from end to end in the supershop management system . A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

The term test scenario and test cases are used interchangeably however the main difference being that test scenarios has several steps however test cases have a single step. When viewed from this perspective test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.

Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects. There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc.

Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing. However, following are the main components which are always available and included in every test case:eg.

- Test case ID.case 001
- Product Module: sales
- Product version : 1
- Revision history : NA
- Purpose : to exercise tests for effective working of sales system
- Assumptions : NA
- Pre-Conditions : available stock and its cost is entered.
- Steps :steps to follow
- Expected Outcome : expected results : such as generation of bill
- Actual Outcome. : actual result such as bill is generated correctly.

- Post Conditions :tendering the remaining amount to customer or generation of statement for the card of the customer.

  Many Test cases can be derived from a single test scenario. In addition to this, some time it happened that multiple test cases are written for single Software which is collectively known as test suites.

  Traceability Matrix

  Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table which is used to trace the requirements during the Software development life Cycle. It can be used for forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). There are many user defined templates for RTM.

  . The main goals for this matrix are:
- Make sure Software is developed as per the mentioned requirements.
- Helps in finding the root cause of any bug.
- Helps in tracing the developed documents during different phases of SDLC.