CS504-Software Engineering-II

(Solved Subjective)

LECTURE FROM

(23 to 45)

Junaidfazal08@gmail.com
Bc190202640@vu.edu.pk

FOR MORE VISIT
VULMSHELP.COME

JUNAID MALIK
0304-1659294

AL-JUNAID TECH INSTITUTE

Question No: 1 (Marks: 2)
What is Software Testing?
Answer:- (Page 192)
To understand the concept of software testing correctly, we need to understand a few related concepts.

Question No: 2 (Marks: 2)
Write unit testing qualitative benefits.
Answer:- (Page 207)
Assessment-oriented: Writing the unit test forces us to deal with design issues - cohesion, coupling.
Confidence-building: We know what works at an early stage. Also easier to change when it's easy to retest.

Question No: 3 (Marks: 2)
Where Term Compute can be used in Methods?
Answer:- (Page 152)
The term compute can be used in methods where something is computed.
valueSet.computeAverage(); matrix.computeInverse()

Question No: 4 (Marks: 2)
Which of the following is a/are non-functional requirement of an e-learning system?

(a) User friendliness
(b) Time taken to download study materials through the system should not be too long.
(c) On-line assignment submission facility
(d) On-line chatting facility
(e) Robustness

Answer:-
(a) User friendliness

Question No: 5 (Marks: 3)
Write unit testing principles.
Answer:-(Page 207)
- In unit testing, developers test their own code units (modules, classes, etc.) during implementation.
- Normal and boundary inputs against expected results are tested.
- Thus unit testing is a great way to test an API.

Question No: 6 (Marks: 3)
The use of *do.... while* loops should be avoided. Explain why ?
Answer:-(Page 159)
The use of *do ... While* loops should be avoided. There are two reasons for this. First is that the construct is superfluous; Any statement that can be written as a *do while* loop can equally well be written as a *while* loop or a *for* loop. Complexity is reduced by minimizing the number of constructs being used. The other reason is of readability. A loop with the conditional part at the end is more difficult to read than one with the conditional at the top.

Question No: 7 (Marks: 3)

Give 3 Equivalence partitioning guidelines

Answer:-(Page 199)

- Organize your equivalence classes. Write them in some order, use some template, sequence, or group them based on their similarities or distinctions. These partitions can be hierarchical or organized in any other manner.
- Boundary conditions: determine boundary conditions. For example, adding in an empty linked list, adding after the last element, adding before the first element, etc.
- You should not forget invalid inputs that a user can give to a system. For example, widgets on a GUI, numeric instead of alphabets, etc.

Question No: 8 (Marks: 5)

Discus the symptoms and an example of memory overrun bug class.

Answer:-(Page 220)

Symptoms

- Program crashes quite regularly after a given routine is called, that routine should be examined for a possible overrun condition.
- If the routine in question does not appear to have any such problem the most likely cause is that another routine, called in the prior sequence, has already trashed variables or memory blocks.
- Checking the trace log of the called routines leading up to one with the problem will often show up the error.

Example

The array only has 50 slots available in its allocation. What happens at that point is that the function goes past the end of the array and starts to walk on things beyond its control. const kMaxEntries = 50;

```
int gArray[kMaxEntries];
char szDummyBuffer[256];
int nState = 10;
int ZeroArray (int *pArray)
{
for (inti=0;i<100;++i)
pArray[i] = 0;
}
```

Question No: 9 (Marks: 5)

Write at leas 5 General Naming conventions for C++ or Java

Answer:-(Page 150)

1. Names representing types must be nouns and written in mixed case starting with upper case.
Line, FilePrefix

2. Variable names must be in mixed case starting with lower case.
line, filePrefix

3. Names representing constants must be all uppercase using underscore to separate words.
MAX_ITERATIONS, COLOR_RED

4. Names representing methods and functions should be verbs and written in mixed case starting with lower case.

getName(), computeTotalWidth()

5. Names representing template types in C++ should be a single uppercase letter.
template<class T>
template<class C, class D>

Question No: 10 (Marks: 5)
Explain at least 2 code structures.
Answer:-(Page 201)
Case:-
In Case statement, control can take either of several branches (as opposed to only two in If statement.) First node represents the switch statement (C/C++) and nodes in middle correspond to all different cases. Program can take one branch and result into the same instruction.

While
A while loop structure consists of a loop guard instruction through which the iteration in the loop is controlled. The control keeps iterating in the loop as long as the loop guard condition is true. It branches to the last instruction when it becomes false.

Question No: 11 (Marks: 2)
What does this mean" Object Creation and Life Time"?
Answer:-(Page 87)
From the object creation and life time point of view, when an object is instantiated, all of its parts must also be instantiated at the same time before any useful work can be done and all of its part die with it. While in the case of association, the life time of two associated object is independent of one another. The only limitation is that an object must be alive or has to be instantiated before a message can be sent to it.

Question No: 12 (Marks: 2)
How one can avoid hazards caused by side effects while writing code. List the two guidelines.
Answer:-(Page 176)
never use "," except for declaration
never use multiple assignments in the same statement

Question No: 13 (Marks: 2)
What is the greatest advantage of exception handling?
Answer:-(Page 184)
One of the most powerful features of exception handling is that an error can be thrown over function boundaries. This allows programmers to put the error handling code in one place, such as the *main*-function of your program.

Question No: 14 (Marks: 2)
Give 2 Unit Testing Tips.
Answer:-(Page 208)
- For small projects you can imbed the unit test for a module in the module itself
- For larger projects you should keep the tests in the package directory or a /test subdirectory of the package

Question No: 15 (Marks: 3)

Write unit testing quantitative benefits.

Answer:-(Page 207)

- **Repeatable**: Unit test cases can be repeated to verify that no unintended side effects have occurred due to some modification in the code.
- **Bounded**: Narrow focus simplifies finding and fixing defects.
- **Cheaper**: Find and fix defects early

Question No: 16 (Marks: 3)

How Comments should be indented relative to their position in the code? Give an example

Answer:-(Page 162)

Comments should be indented relative to their position in the code.

```
while (true) {      // NOT: while (true) {
// Do something  // // Do something
something();      //     something();
}                 //     }
```

Question No: 17 (Marks: 3)

Consider the following code fragment.

```
while a
{
while b
c
d
}
```

If you were to test this code, what would be the test technique to adopt?

Question No: 18 (Marks: 5)

Narrate the manner for the organization of Class and Interface declarations

Answer:-(Page 157)

Class and Interface declarations should be organized in the following manner:

1. Class/Interface documentation.
2. Class or interface statement.
3. Class (**static**) variables in the order **public**, **protected**, package (no access modifier), **private**.
4. Instance variables in the order **public**, **protected**, package (no access modifier), private.
5. Constructors.
6. Methods (no specific order).

Question No: 19 (Marks: 5)

Discus the symptoms and an example of coding error bug class.

Answer:-(Page 219)

Symptoms

- Unexpected errors in black box testing.
- The errors that unexpectedly occur are usually caused by coding errors.
- Compiler warnings.
- Coding errors are usually caused by lack of attention to details.

Example

In the following example, a function accepts an input integer and converts it into a string that contains that integer in its word representation. void convertToString(int InInteger,

char* OutString, int* OutLength)
{
switch(InInteger){
case 1: OutString = "One";OutLength = 3;
break;
case 2: OutString = "Two";OutLength = 3;
break;
case 3: OutString = "Three";OutLength = 5;
break;
case 4: OutString = "Four";OutLength = 4;
break;
case 5: OutString = "Five";OutLength = 4;
break;
case 6: OutString = "Six";OutLength = 3;
break;
case 7: OutString = "Seven";OutLength = 5;
break;
case 8: OutString = "Eight";OutLength = 5;
break;
case 9: OutString = "Nine";OutLength = 4;
break;
}
}

Question No: 20 (Marks: 5)

Why Code portability is so important? Give out 3 ways / Guide lines to improve the code portability with examples (5+5)

Answer:-(Page 179)

Many applications need to be ported on to many different platforms. As we have seen, it is pretty hard to write error free, efficient, and maintainable software. So, if a major rework is required to port a program written for one environment to another, it will be probably not come at a low cost. So, we ought to find ways and means by which we can port applications to other platforms with minimum effort. The key to this lies in how we write our program. If we are careful during writing code, we can make it portable. On the other hand if we write code without portability in mind, we may end-up with a code that is extremely hard to port to other environment. Following is brief guideline that can help you in writing portable code.

Stick to the standard

1. Use ANSI/ISO standard C++
2. Instead of using vendor specific language extensions, use STL as much as possible

Program in the mainstream

Although C++ standard does not require function prototypes, one should always write them. double sqrt(); // old style acceptable by ANSI C double sqrt(double); // ANSI – the right approach

Size of data types

Sizes of data types cause major portability issues as they vary from one machine to the other so one should be careful with them.

```
int i, j, k;
…
j = 20000;
k = 30000;
i = j + k;
// works if int is 4 bytes
// what will happen if int is 2 bytes?
```

Question No: 21 (Marks: 2)

What is an Inspection Checklist?

Answer:-(Page 210)

Checklist of common errors in a program should be developed and used to drive the inspection process. These error checklists are programming language dependent such that the inspector has to analyze major constructs of the programming language and develop checklists to verify code that is written using these checklists.

Question No: 22 (Marks: 2)

Give 2 examples of exceptional code pathes.

Answer:-(Page 186)

- if (e.Title() == "CEO" || e.Salary() > 10000)
  operator ==() might throw.
- cout << e.First() << " " << e.Last() << " is overpaid" << endl;
  As per C++ standard, any of the five calls to << operator might throw.

Question No: 23 (Marks: 2)

The following written statement depicts which requirement from the requirement engineering process "Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc."

Question No. 24

What do we mean by ambiguous requirements. Explain with the help of a example.

Answer:- (Page 198)

Ambiguity means that two different readers of the same document interpret the requirement differently. Ambiguity arises from the use of natural language. Because of the imprecise nature of the language, different readers interpret the statements differently. As an example, consider the following Urdu Phrase: ***"Rooko mut jane doo".*** Now, depending upon where a reader places the comma in this statement, two different readers may interpret it in totally different manner. If a comma is palced after ***"Rooko"***, the sentence will become ***"Rooko, mut jane doo",*** meaning ***"don't let him go".*** On the other hand if the comma id placed after ***"mut",*** the

sentence will become ***"Rooko mut, jane doo",*** meaning ***"let him go".*** Ambiguous requirements therefore result in misunderstandings and mismatched expectations, resulting in a wasted time and effort and an undesirable product.

Question No: 25 (Marks: 3)

Give three general rules for avoiding split lines.

Answer:-(Page 155)

for avoiding split lines don't use these

- Break after a comma.
- Break after an operator.
- Align the new line with the beginning of the expression on the previous line.

Question No: 26 (Marks: 3)

Explain about 3 coverage schemes in white box testing.

Answer:-(Page 202)

Statement Coverage: In this scheme, statements of the code are tested for a successful test that checks all the statements lying on the path of a successful scenario.

Branch Coverage: In this scheme, all the possible branches of decision structures are tested. Therefore, sequences of statements following a decision are tested.

Path Coverage: In path coverage, all possible paths of a program from input instruction to the output instruction are tested. An exhaustive list of test cases is generated and tested against the code.

Question No: 27 (Marks: 5)

List five guidelines that can help you in writing portable code.

Answer:-(Page 179)

Stick to the standard

1. Use ANSI/ISO standard C++
2. Instead of using vendor specific language extensions, use STL as much as possible.

Program in the mainstream

Although C++ standard does not require function prototypes, one should always write them.
double sqrt(); // old style acceptable by ANSI C double sqrt(double); // ANSI – the right approach

Size of data types

Sizes of data types cause major portability issues as they vary from one machine to the other so one should be careful with them.

```
int i, j, k;
…
j = 20000;
k = 30000;
i = j + k;
// works if int is 4 bytes
// what will happen if int is 2 bytes?
```

Order of Evaluation

As mentioned earlier during the discussion of side effects, order of evaluation varies from one implementation to other. This therefore also causes portability issues. We should therefore follow guidelines mentioned in the side effect discussion.

Arithmetic or Logical Shift

The C/C++ language has not specified whether right shift >> is arithmetic or logical. In the arithmetic shift sign bit is copied while the logical shift fills the vacated bits with 0. This obviously reduces portability. Interestingly, Java has introduced a new operator to handle this issue. >> is used for arithmetic shift and >>> for logical shift.

QuestionNo:28 (marks 5)

Below is the chunk of code :

Result=squareRoot(arget;

Assert (abs (result * result – argument) < epsilon);

Write the Contract for square root routine keeping in view unit testing.

Answer:-(Page 207)

- Pass in a negative argument and ensure that it is rejected
- Pass in an argument of zero to ensure that it is accepted (this is a boundary value)
- Pass in values between zero and the maximum expressible argument and verify that the difference between the square of the result and the original argument is less than some value epsilon.

Question No: 29 (Marks: 5)

Parentheses should always be used as they reduce complexity. Explain it with the help of a single example.

Answer:-(Page 163)

Parentheses should always be used as they reduce complexity and clarify things by specifying grouping. It is especially important to use parentheses when different unrelated operators are used in the same expression as the precedence rules are often assumed by the programmers, resulting in logical errors that are very difficult to spot. As an example consider the following statement:

if (x & MASK == BITS)

This causes problems because == operator has higher precedence than & operator. Hence, MASK and BITS are first compared for equality and then the result, which is 0 or 1, is added with x. This kind of error will be extremely hard to catch. If, however, parentheses are used, there will be no ambiguity as shown below. if ((x & MASK) == BITS)

Question No: 30 (Marks: 2)

Define Modularity.

Answer:-(Page 170)

Modularity is a tool that can help us in reducing the size of individual functions, making them more readable.

Question No: 31 (Marks: 2)

Differentiate between Architectural Design and System Architecture in one line.

Answer:-

Architecture faces towards strategy, structure and purpose, towards the abstract while Design faces towards implementation and practice, towards the concrete.

AL-JUNAID TECH INSTITUTE

Question No: 32 (Marks: 2)

What is meant by Software Debugging?

Answer:-(Page 214)

Debugging techniques are the only mechanism to reach at the code that is malfunctioning.

Question No.33

Differentiate between Black box testing and white box testing.

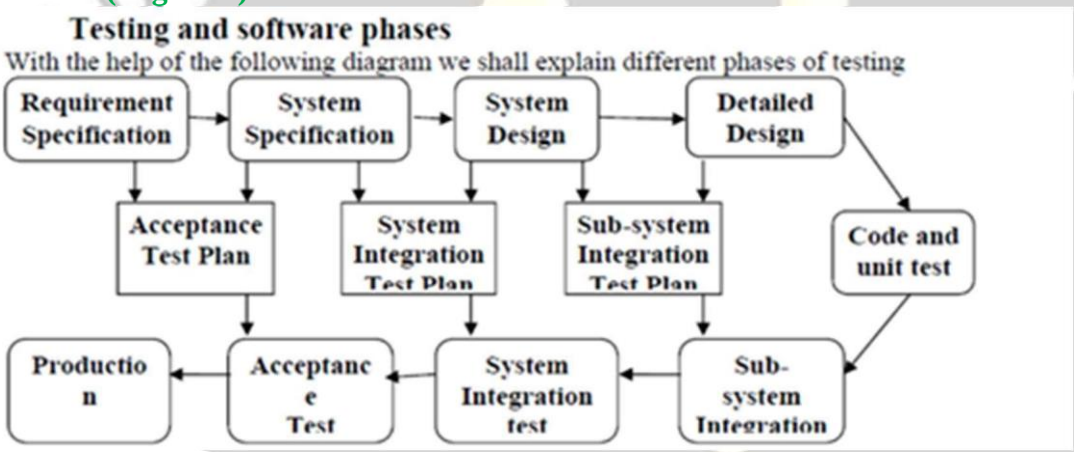
Answer:- (Page 198)

- In this type of testing, a component or system is treated as a black box and it is tested for the required behavior. This type of testing is not concerned with how the inputs are transformed into outputs.
- As opposed to black box testing, in structural or white box testing we look inside the system and evaluate what it consists of and how is it implemented.

Question No: 34 (Marks: 3)

What are the different phases of testing? Draw a diagram. (No write up required)

Answer:-(Page 197)



Question No: 35 (Marks: 5)

Why and how complex expressions should be written in multiple short statements, Explain it with example

Answer:-(Page 164)

Complex expressions should be broken down into multiple statements. An expression is considered to be complex if it uses many operators in a single statement. As an example consider the following statement:

```
*x += (*xp=(2*k < (n-m) ? c[k+1] : d[k--]));
```

This statement liberally uses a number of operators and hence is very difficult to follow and understand. If it is broken down into simple set of statements, the logic becomes easier to follow as shown below:

```
if (2*k < n-m)
*xp = c[k+1];

else
*xp = d[k--];
```

*x = *x + *xp;

Question No: 36 (Marks: 5)

What is the Usefulness of Testing?

Answer:-(Page 197)

Objective of testing is to discover and fix as many errors as possible before the software is put to use. That is before it is shipped to the client and the client runs it for acceptance. In software development organizations, a rift exists between the development and the testing teams. Often developers are found questioning about the significance or even need to have the testing resources in the project teams. Whoever doubts on the usefulness of the testing team should understand what could happen if the application is delivered to client without testing? At the best, the client may ask to fix all the defects (free of cost) he would discover during the acceptance testing. At the worst, probably he would sue the development firm for damages. However, in practice, clients are often seen complaining about the deliverables and a couple of defected deliverables are sufficient for breaking the relations next to the cancellation of contract.

Question No: 37 (Marks: 5)

What are the static analyzers, give a check list of the requirements?

Answer:-(Page 211)

Static analyzers are software tools for source text processing. They parse the program text and try to discover potentially erroneous conditions and bring these to the attention of the verification and validation team.

Checklist for static analysis

Data faults	• variable used before initialization • variable declared but never used • variables assigned twice but never used between assignments • possible array bound violations • undeclared variables
Control faults	• unreachable code • unconditional branches into loops
Input/Output faults	• variable output twice with no intervening assignment
Storage Management fault	• unassigned pointers • pointer arithmetic

Question No: 38 (Marks: 2)

What is called self documenting code?

Answer:-(Page 147)

A self documenting code is a code that explains itself without the need of comments and extraneous documentation, like

- Flow charts,
- UML diagrams,
- Process-flow state

Question No: 39 (Marks: 2)

What are the six elements that are present in every computer-based system?

Answer:

1. Hardware
2. People/user

3. Data
4. Procedure
5. Connectivity
6. software

Question No. 40

Consider the following Use Case diagram: Identify the system actors in given use case diagram.

Question No: 41 (Marks: 2)

What are the true statements with respect to equivalence partitioning?

(a) Input data and output results often fall into different classes where all members of a class are related.
(b) It is a method to partition the requirements into equivalent classes during the requirement analysis process.
(c) Test cases should be chosen to be representative of each equivalence partition.
(d) It is recommended that only boundaries are checked in each partition.
(e) It is recommended that boundaries as well as mid points are checked in each partition.

Answer:-

(A ,b ,c,) are the true statements with respect to equivalence partitioning

Question No. 42

Bit fields do suffer from a lack of portability between platforms. Why?

Bit fields are a convenient way to express many difficult operations. However, bit fields do suffer from a lack of portability between platforms:

Answer:- (Page 183)

- integers may be signed or unsigned
- Many compilers limit the maximum number of bits in the bit field to the size of an integer which may be either 16-bit or 32-bit varieties.
- Some bit field members are stored left to right others are stored right to left in memory.
- If bit fields too large, next bit field may be stored consecutively in memory (overlapping the boundary between memory locations) or in the next word of memory.

Question No: 43 (Marks: 3)

Why Special characters like TAB and page break must be avoided? Explain

Answer: - (Page 155)

Special characters like TAB and page break must be avoided. These characters are bound to cause problem for editors, printers, terminal emulators or debuggers when used in a multi-programmer, multi-platform environment

Question No.44

Define these terms: Branch Coverage, Statement Coverage.

Answer:- (Page 202)

Branch Coverage: In this scheme, all the possible branches of decision structures are tested. Therefore, sequences of statements following a decision are tested.

Statement Coverage: In this scheme, statements of the code are tested for a successful test that checks all the statements lying on the path of a successful scenario.

Question No: 45 (Marks: 5)

What is the Software testing objective? Also define a successful test.

Answer: - (Page 193)

Software testing objective

- The correct approach to testing a scientific theory is not to try to verify it, but to seek to refute the theory. That is to prove that it has errors. (Popper 1965)
- The goal of testing is to expose latent defects in a software system before it is put to use.
- A software tester tries to break the system. The objective is to show the presence of a defect not the absence of it.
- Testing cannot show the absence of a defect. It only increases your confidence in the software.
- This is because exhaustive testing of software is not possible – it is simply too expansive and needs virtually infinite resources.

Successful Test

From the following sayings, a successful test can be defined "If you think your task is to find problems then you will look harder for them than if you think your task is to verify that the program has none" – Myers 1979.

"A test is said to be successful if it discovers an error" – doctor's analogy.

The success of a test depends upon the ability to discover a bug not in the ability to prove that the software does not have one. As, it is impossible to check all the different scenarios of a software application, however, we can apply techniques that can discover potential bugs from the application. Thus a test that helps in discovering a bug is a successful test. In software testing phase, our emphasis is on discovering all the major bugs that can be identified by running certain test scenarios. However it is important to keep in mind that testing activity has certain limitations.

Question No.46

Memory over-runs and their symptoms

Answer:- (Page 220)

A memory overrun occurs when you use memory that does not belong to you. (**Symptoms repeated**)

Question No.47

Identifier names also play a significant role

Answer:- (Page 148)

Identifier names also play a significant role in enhancing the readability of a program. The names should be chosen in order to make them meaningful to the reader. In order to understand the concept, let us look at the following statement.

if (x==0) // this is the case when we are allocating a new number

In this particular case, the meanings of the condition in the if-statement are not clear and we had to write a comment to explain it. This can be improved if, instead of using x, we use a more meaningful name. Our new code becomes:

if (AllocFlag == 0)

The situation has improved a little bit but the semantics of the condition are still not very clear as the meaning of 0 is not very clear. Now consider the following statement:

If (AllocFlag == NEW_NUMBER)

We have improved the quality of the code by replacing the number 0 with a named constant NEW_NUMBER. Now, the semantics are clear and do not need any extra comments, hence this piece of code is self-documenting.

Question No. 48

Good clues, Easy Bugs
explain this term.

Answer:- (Page 226)

Good clues, Easy Bugs
Get A Stack Trace
In the debugging process a stack trace is a very useful tool.
Following stack trace information may help in debugging process.

- Source line numbers in stack trace is the single, most useful piece of debugging information.
- After that, values of arguments are important
  - Are the values improbable (zero, very large, negative, character strings with non-alphabetic characters?
- Debuggers can be used to display values of local or global variables.
  - These give additional information about what went wrong.

Question No. 49

What is the syntax used for naming objects in a sequence diagrams?

Answer:- (Page 107)

The syntax used for naming objects in a sequence diagram is as follows:

- syntax: [instanceName][:className]
- Name classes consistently with your class diagram (same classes).
- Include instance names when objects are referred to in messages or when several objects of the same type exist in the diagram.

Question No: 50

Define Textual Analysis? 3 Numbers

Answer:- (Page 90)

The oldest techniques to identify objects and their relationships. This technique is called Textual Analysis. It was initially developed by Abbot and then extended by Graham and others. In this technique different parts of speech are identified within the text of the specification and these parts are modeled using different components.

AL-JUNAID TECH INSTITUTE

Question No: 51

Describe Loop Errors and symptoms? 5 Numbers

Answer:- (Page 220)

Loop Errors

- Loop errors break down into several different subtypes.
- They occur around a loop construct in a program.
- Infinite loops, off-by-one loops, and improperly exited loops.

Symptoms

- If your program simply locks up, repeatedly displays the same data over and over, or infinitely displays the same message box, you should immediately suspect an infinite loop error.
- Off-by-one loop errors are quite often seen in processes that perform calculations.
- If a hand calculation shows that the total or final sum is incorrect by the last data point, you can quickly surmise that an off-by-one loop error is to blame.
- Likewise, if you were using graphics software and saw all of the points on the screen, but the last two were unconnected, you would suspect an off-by-one error.

Question No: 52

Write Note on Partitioning? 5 Numbers

Answer:- (Page 135)

Partitioning of architecture is an important concept. What we basically want to do is distribute the responsibilities to different subsystems so that we get a software system which is easy to maintain. Partitioning results in a system that suffers from fewer side effects. This ultimately means that we get a system that is easier to test and extend and hence is easier to maintain.

Partitioning of architecture may be “horizontal” and/or “vertical”.

In the horizontal partitioning we define separate branches of the module hierarchy for each major function and control modules are used to coordinate communication between functions.

Vertical partitioning divides the application from a decision making perspective. The architecture is partitioned in horizontal layers so that decision making and work are stratified with the decision making modules residing at the top of the hierarchy and worker coming at the bottom. This partitioning is also known as factoring.

Question No: 53

How software engineer can work on domains if he gets the technical work..?

Answer:- (Page 5)

“All aspects of software production’ Software engineering is not just concerned with the technical processes of software development but also with activities such as software project management and with the development of tools, methods and theories to support software production”.
These definitions make it clear that Software Engineering is not just about writing code.

Question No:54

equivalence classes

Answer:- (Page 199)

Equivalence Classes or Equivalence Partitioning

Two tests are considered to be equivalent if it is believed that:

if one discovers a defect, the other probably will too, and
If one does not discover a defect, the other probably won't either.
Equivalence classes help you in designing test cases to test the system effectively and efficiently. One should have reasons to believe that the test cases are equivalent.

Question No: 55

Memory and resource leak symptoms 3 marks

Answer:- (Page 216)

Symptoms

- System slowdowns
- Crashes that occur "randomly" over a long period of time

Question No: 56

Tools and Methods used in software engineering

Answer:- (Page 6 & 11)

- Programming Language
- Programming Language Design
- Software Design Techniques
- Tools
- Testing
- Maintenance
- Development etc.

Methods: Methods provide the technical "how-to's" to carryout these tasks. There could be more than one technique to perform a task and different techniques could be used in different situations.
Tools: Tools provide automated or semi-automated support for software processes, methods, and quality control.

Question No: 57

Five guidelines using for Code

Answer:- (Page 150)

1. Names representing types must be nouns and written in mixed case starting with upper case.
Line, FilePrefix

2. Variable names must be in mixed case starting with lower case. line, filePrefix

3. Names representing constants must be all uppercase using underscore to separate words.
MAX_ITERATIONS, COLOR_RED

4. Names representing methods and functions should be verbs and written in mixed case starting with lower case.
getName(), computeTotalWidth()

5. Names representing template types in C++ should be a single uppercase letter.
template<class C, class D>

Question No: 58

Inspection pre-conditions?

Answer:- (Page 210)

A precise specification must be available before inspections. Team members must be familiar with the organization standards. In addition to it, syntactically correct code must be available to the inspectors. Inspectors should prepare a checklist that can help them during the inspection process.

Question No: 59

WHITE BOX TESTING?

Answer:- (Page 202)

In white box testing we test the structure of the program. In this technique the test cases are written in a manner to cover different possibilities in code.

Question No: 60

Software development processes?

Answer:- (Page 8)

Software development is a process of balancing among different characteristics of software described in the previous section. And it is an art to come up with such a good balance and that art can be learned from e

Question No: 61

INCLUDE FILES AND INCLUDE STATEMENT?

Answer:- (Page 157)

1. Header files must include a construction that prevents multiple inclusions. The convention is an all uppercase construction of the module name, the file name and the h
Suffix.

```
#ifndef MOD_FILENAME_H
#define MOD_FILENAME_H
:
#endif
```

The construction is to avoid compilation errors. The construction should appear in the top of the file (before the file header) so file parsing is aborted immediately and compilation time is reduced.

xperience.

Question No: 62

BALANCING ACT IN SOFTWARE ENGINEERING?

Answer:- (Page 7)

Software Engineering is actually the balancing act. You have to balance many things like cost, user friendliness, Efficiency, Reliability etc. You have to analyze which one is the more important feature for your software is it reliability, efficiency, user friendliness or something else.

Question No: 63

THREE NON EXCEPTIONAL PATHS?

Answer:- (Page 185)

- if (e.Title() == “CEO” || e.Salary() > 10000)
  - · if e.Title() == “CEO” is true then the second part is not evaluated and e.Salary() will not be called.
  - · cout will be performed
-

if e.Title() != “CEO” and e.Salary() > 10000

· both parts of the condition will be evaluated

· cout will be performed.

❖

if e.Title() != “CEO” and e.Salary() <= 10000

· both parts of the condition will be evaluated

· cout will not be performed.