Uni. RN													

UNITED COLLEGE OF ENGINEERING & RESEARCH, PRAYAGRAJ (010)		**Department CSE & IT**	
Second Sessional Examination (Even Semester 2021-22)		SEMESTER: VI	Date:- 09/05/22
TIME: 2 hours.	SUBJECT : Software Engineering	Paper code: KCS 601	MM. 30

READ ALL INSTRUCTIONS AND QUESTIONS VERY CAREFULLY

SECTION A (Attempt ALL questions) Very short answer			**[10]**	**CO**	**Bloom's Taxonomy Level**
1	a	List major software design tools. Data flow diagram, E-R diagram, and Use case diagram	[1]	3	L1
1	b	State the differences between top-down and bottom up design approaches. While the top-down approach focuses on breaking down a big problem into smaller and understandable chunks, the bottom-up approach first focuses on solving the smaller problems at the fundamental level and then integrating them into a whole and complete solution.	[1]	3	L1
1	c	Discuss the statement "common coupling is the worst type of coupling between two modules". The "content" coupling not common coupling is the worst type of coupling and it should be avoided.	[1]	3	L2
1	d	Differentiate between flow chart and structure chart. Flow chart is a convenient technique to represent the flow of control in a program. Structure chart represents the software architecture.	[1]	3	L4
1	e	Define control flow graph? State the expression for calculating cyclomatic complexity. A Control Flow Graph (CFG) is the graphical representation of control flow or computation during the execution of programs or applications. The Cyclomatic complexity, V (G) for a flow graph G can be defined as V (G) = E - N + 2.	[1]	3	L1, L1
1	f	Differentiate between function oriented design and object oriented design? Both Functional programming and object-oriented programming uses a different method for storing and manipulating the data. In functional programming, data cannot be stored in objects and it can only be transformed by creating functions. In object-oriented programming, data is stored in objects.	[1]	3	L4
1	g	State the expression for token count, length of the program, program volume, program level given by halestead's software science metrics. Token count(n)= n1+n2, Program length(N)=N1+N2, Program Volume(V)=N*log2 n, Program level(L)=1/D.	[1]	3	L1
1	h	Define equivalence class partitioning. Equivalence partitioning or equivalence class partitioning (ECP) is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.	[1]	4	L1
1	i	Describe black-box testing. Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.	[1]	4	L2
1	j	Differentiate between static testing and dynamic testing. Static testing covers the structural and statement coverage testing. Dynamic testing covers the executable file of the code. It consists of reviews, walkthrough, inspection and static analysis of code. It consists of functional testing, non-functional testing and data/control flow analysis.	[1]	4	L4
SECTION B (Attempt Any three questions) Short answer			**[12]**		

Uni. RN													

2		Explain the following: a) Acceptance testing It is a formal testing according to user needs, requirements and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers or other authorized entities to determine whether to accept the system or not. b) Regression testing Regression testing is a type of testing where you can verify that the changes made in the codebase do not impact the existing software functionality. For example, these code changes could include adding new features, fixing bugs, or updating a current feature.	[4]	4	L2
3		Define boundary value analysis? Consider a program for the determination of the nature of roots of a quadratic equation. Its input is a triple of positive integers (say a, b, c) and values may be from interval [0, 100]. The program output may have one of the following words. [Not a quadratic equation; Real roots; Imaginary roots; Equal roots] Discuss the boundary value test cases. Boundary-value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. The boundary value test cases are : *Test Case* | *a* | *b* | *c* | *Expected output* 1 | 0 | 50 | 50 | Not Quadratic 2 | 1 | 50 | 50 | Real Roots 3 | 50 | 50 | 50 | Imaginary Roots 4 | 99 | 50 | 50 | Imaginary Roots 5 | 100 | 50 | 50 | Imaginary Roots 6 | 50 | 0 | 50 | Imaginary Roots 7 | 50 | 1 | 50 | Imaginary Roots 8 | 50 | 99 | 50 | Imaginary Roots 9 | 50 | 100 | 50 | Equal Roots 10 | 50 | 50 | 0 | Real Roots 11 | 50 | 50 | 1 | Real Roots 12 | 50 | 50 | 99 | Imaginary Roots 13 | 50 | 50 | 100 | Imaginary Roots	[4]	4	L1, L2
4		Describe unit testing. Explain test drivers and test stubs and examine its importance for unit testing. Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. **Test Driver** – the dummy modules that simulate the high level modules. **Test Stub** – the dummy modules that simulates the low level modules. The importance of designing drivers and stubs is that a unit will work/behave in the simulated environment as in the actual software environment. Now a unit test can be written against the interface and the unit will still work properly once the drivers and stubs have been placed correctly.	[4]	4	L2, L2, L4
5		Explain white-box testing. Discuss the various strategies for white box testing. White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems. The following are some important techniques of white-box testing: A. Basis Path Testing The test cases created from the basis path allow the program to be	[4]	4	L2, L2

Uni. RN													

		executed in such a way as to examine each possible path through the program by executing each statement at least once B. Structural Testing It is a broaden testing coverage and improve quality of white-box testing.The following are some important types of structural testing: 1. Statement coverage testing 2. Branch coverage testing 3. Condition coverage testing 4. Path coverage testing 5. Data flow based testing C. Logic Based Testing It is used when the input domain and resulting processing are amenable to a **decision table representation.**			
SECTION C (Attempt ANY ONE question) Long answer			**[8]**		
6		Explain the need for software measures and describe various metrics. **Need of Software Measurement:** Software is measured to: 1. Create the quality of the current product or process. 2. Anticipate future qualities of the product or process. 3. Enhance the quality of a product or process. 4. Regulate the state of the project in relation to budget and schedule. Software metrics:- LOC Metrics It is one of the earliest and simpler metrics for calculating the size of the computer program. It is generally used in calculating and comparing the productivity of programmers. Halstead's Software Metrics According to Halstead's "A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operand." Token Count In these metrics, a computer program is considered to be a collection of tokens, which may be classified as either operators or operands. All software science metrics can be defined in terms of these basic symbols. These symbols are called as a token. The basic measures are n1 = count of unique operators. n2 = count of unique operands.	[8]	3	L2, L2

N1 = count of total occurrences of operators.
N2 = count of total occurrence of operands.

Halstead metrics are:

Program Volume (V)

The unit of measurement of volume is the standard unit for size "bits." It is the actual size of a program if a uniform binary encoding for the vocabulary is used.

$V=N*\log_2 n$

Program Level (L)

The value of L ranges between zero and one, with L=1 representing a program written at the highest possible level (i.e., with minimum size).

$L=V^*/V$

Program Difficulty

The difficulty level or error-proneness (D) of the program is proportional to the number of the unique operator in the program.

$D= (n1/2) * (N2/n2)$

Programming Effort (E)

The unit of measurement of E is elementary mental discriminations.

$E=V/L=D*V$

Estimated Program Length

estimated program length is denoted by N^

$N^ = n1\log_2 n1 + n2\log_2 n2$

Functional Point (FP) Analysis

Allan J. Albrecht initially developed function Point Analysis in 1979 at IBM and it has been further modified by the International Function Point Users Group (IFPUG). FPA is used to make estimate of the software project, including its testing in terms of functionality or function size of the software product.

FP = Count-total * [0.65 + 0.01 * $\sum(f_i)$]
= Count-total * CAF

where Count-total is obtained from the above Table.

CAF = [0.65 + 0.01 *$\sum(f_i)$]

Uni. RN													

7		State the characteristics of a good design? Describe different types of coupling and cohesion. How design evaluation is performed? Six characteristics of good software design—**simplicity, coupling, cohesion, information hiding, performance, and security.** **Types of Coupling:** • **Data Coupling:** If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled. In data coupling, the components are independent of each other and communicate through data. Module communications don't contain tramp data. Example-customer billing system. • **Stamp Coupling** In stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves tramp data. It may be necessary due to efficiency factors- this choice was made by the insightful designer, not a lazy programmer. • **Control Coupling:** If the modules communicate by passing control information, then they are said to be control coupled. It can be bad if parameters indicate completely different behavior and good if parameters allow factoring and reuse of functionality. Example- sort function that takes comparison function as an argument. • **External Coupling:** In external coupling, the modules depend on other modules, external to the software being developed or to a particular type of hardware. Ex- protocol, external file, device format, etc. • **Common Coupling:** The modules have shared data such as global data structures. The changes in global data mean tracing back to all modules which access that data to evaluate the effect of the change. So it has got disadvantages like difficulty in reusing modules, reduced ability to control data accesses, and reduced maintainability. • **Content Coupling:** In a content coupling, one module can modify the data of another module, or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided. **Types of Cohesion:** • **Functional Cohesion:** Every essential element for a single computation is contained in the component. A functional cohesion performs the task and functions. It is an ideal situation. • **Sequential Cohesion:** An element outputs some data that becomes the input for other element, i.e., data flow between the parts. It occurs naturally in functional programming languages. • **Communicational Cohesion:** Two elements operate on the same input data or contribute towards the same output data. Example- update record in the database and send it to the printer. • **Procedural Cohesion:** Elements of procedural cohesion ensure the order of execution. Actions are still weakly connected and unlikely to be reusable. Ex- calculate student GPA, print student record, calculate cumulative GPA, print cumulative GPA. • **Temporal Cohesion:** The elements are related by their timing involved. A module connected with temporal cohesion all the tasks must be executed in the same time span. This cohesion contains the code for initializing all the parts of the system. Lots	[8]	3	L1, L2

Uni. RN													

		of different activities occur, all at unit time. • **Logical Cohesion:** The elements are logically related and not functionally. Ex- A component reads inputs from tape, disk, and network. All the code for these functions is in the same component. Operations are related, but the functions are significantly different. • **Coincidental Cohesion:** The elements are not related(unrelated). The elements have no conceptual relationship other than location in source code. It is accidental and the worst form of cohesion. Ex- print next line and reverse the characters of a string in a single component. **Design evaluation-** ➢ In evaluating the software Design, many factors are used, out of which two important factors are – ▪ Coupling ▪ Cohesion **A good design process should aim at reducing coupling.** **A good design process should try to maximize cohesion of each module.**			
####	**END OF PAPER**	**####**			

Course Outcome Wise Marks Distribution	CO1	CO2	CO3	CO4	CO5
			23	19	

Bloom's Taxonomy Wise Marks Distribution	L1	L2	L3	L4	L5	L6
	11	26		5		