

Internal Assessment Test - II

Sub:	SOFTWARE ENGINEERING	Code:	15CS42
Date:	08 / 05 / 2017	Duration:	90 mins
		Max Marks:	50
		Sem:	4(A,B,C,D)
		Branch:	CSE

Answer **FOUR FULL** questions selecting **AT LEAST ONE** question **from each part**

	Marks	OBE																																																					
		CO	RBT																																																				
<u>PART A</u>																																																							
1 (a) A program specification states that the program accepts 4 to 10 inputs that are five-digit integers greater than or equal to 10,000. Show equivalence partitions and possible test input values.	[5]	CO5	L3																																																				
(b) Explain Requirements-based testing with the help of an example.	[5]	CO5	L3																																																				
OR																																																							
2 (a) Explain various interface types and interface errors.	[8]	CO5	L2																																																				
(b) Differentiate between verification and validation.	[2]	CO5	L1																																																				
<u>PART B</u>																																																							
3 (a) Explain testing process with the help of a neat diagram.	[8]	CO5	L2																																																				
(b) Define test-driven development.	[2]	CO5	L1																																																				
OR																																																							
4 (a) Explain software re-engineering with the help of a neat diagram.	[6]	CO4	L2																																																				
(b) Explain maintenance cost factors that result in high cost of maintenance.	[4]	CO5	L2																																																				
<u>PART C</u>																																																							
5 (a) With the help of a neat diagram, explain software evolution process.	[7]	CO5	L2																																																				
(b) Define three types of maintenance.	[3]	CO5	L2																																																				
OR																																																							
6 (a) What is program evolution dynamics? Explain Lehman laws.	[2+8]	CO5	L2																																																				
<u>PART D</u>																																																							
7 (a) Differentiate between milestones and deliverables.	[2]	CO5	L1																																																				
(b) Draw a neat diagram to explain project planning process.	[8]	CO4	L2																																																				
(c) Explain COCOMO-II model with a neat diagram	[10]	CO4	L2																																																				
OR																																																							
8 From the information given in the table, draw																																																							
(a) Bar chart/Gantt chart	[10]	CO4	L3																																																				
(b) Staff allocation chart	[10]	CO4	L3																																																				
<table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th>Task</th> <th>Effort (person-days)</th> <th>Duration (days)</th> <th>Dependencies</th> </tr> </thead> <tbody> <tr><td>T1</td><td style="text-align: center;">15</td><td style="text-align: center;">10</td><td style="text-align: center;">-</td></tr> <tr><td>T2</td><td style="text-align: center;">8</td><td style="text-align: center;">15</td><td style="text-align: center;">-</td></tr> <tr><td>T3</td><td style="text-align: center;">20</td><td style="text-align: center;">15</td><td style="text-align: center;">T1 (M1)</td></tr> <tr><td>T4</td><td style="text-align: center;">5</td><td style="text-align: center;">10</td><td style="text-align: center;">-</td></tr> <tr><td>T5</td><td style="text-align: center;">5</td><td style="text-align: center;">10</td><td style="text-align: center;">T2, T4 (M3)</td></tr> <tr><td>T6</td><td style="text-align: center;">10</td><td style="text-align: center;">5</td><td style="text-align: center;">T1, T2 (M4)</td></tr> <tr><td>T7</td><td style="text-align: center;">25</td><td style="text-align: center;">20</td><td style="text-align: center;">T1 (M1)</td></tr> <tr><td>T8</td><td style="text-align: center;">75</td><td style="text-align: center;">25</td><td style="text-align: center;">T4 (M2)</td></tr> <tr><td>T9</td><td style="text-align: center;">10</td><td style="text-align: center;">15</td><td style="text-align: center;">T3, T6 (M5)</td></tr> <tr><td>T10</td><td style="text-align: center;">20</td><td style="text-align: center;">15</td><td style="text-align: center;">T7, T8 (M6)</td></tr> <tr><td>T11</td><td style="text-align: center;">10</td><td style="text-align: center;">10</td><td style="text-align: center;">T9 (M7)</td></tr> <tr><td>T12</td><td style="text-align: center;">20</td><td style="text-align: center;">10</td><td style="text-align: center;">T10, T11 (M8)</td></tr> </tbody> </table>				Task	Effort (person-days)	Duration (days)	Dependencies	T1	15	10	-	T2	8	15	-	T3	20	15	T1 (M1)	T4	5	10	-	T5	5	10	T2, T4 (M3)	T6	10	5	T1, T2 (M4)	T7	25	20	T1 (M1)	T8	75	25	T4 (M2)	T9	10	15	T3, T6 (M5)	T10	20	15	T7, T8 (M6)	T11	10	10	T9 (M7)	T12	20	10	T10, T11 (M8)
Task	Effort (person-days)	Duration (days)	Dependencies																																																				
T1	15	10	-																																																				
T2	8	15	-																																																				
T3	20	15	T1 (M1)																																																				
T4	5	10	-																																																				
T5	5	10	T2, T4 (M3)																																																				
T6	10	5	T1, T2 (M4)																																																				
T7	25	20	T1 (M1)																																																				
T8	75	25	T4 (M2)																																																				
T9	10	15	T3, T6 (M5)																																																				
T10	20	15	T7, T8 (M6)																																																				
T11	10	10	T9 (M7)																																																				
T12	20	10	T10, T11 (M8)																																																				

Internal Assessment Test – II Scheme of Evaluation

Sub:	SOFTWARE ENGINEERING	Code:	15CS42
Date:	08 / 05 / 2017	Duration:	90 mins
		Max Marks:	50
		Sem:	4(A,B,C)
		Branch:	CSE

Answer **FOUR FULL** questions selecting **AT LEAST ONE** question **from each part**

		Marks	OBE	
			CO	RBT
<u>PART A</u>				
1 (a)	<p>A program specification states that the program accepts 4 to 10 inputs that are five-digit integers greater than or equal to 10,000. Show equivalence partitions and possible test input values.</p> <p>No. of inputs: (2.5M) Less than 4: 3 Between 4 and 10: 4, (5 or 6 or 7 or 8 or 9), 10 Greater than 10: 11</p> <p>Input values: (2.5M) Less than 10000: 9999 Between 10000 and 99999: 10000, 50000 (or any other), 99999 Greater than 99999: 100000</p>	[5]	CO5	L3
(b)	<p>Explain Requirements-based testing with the help of an example. Requirements based testing: (1M definition, 4M example)</p> <p>(i) Requirements-based testing - Examining each req. & developing test for it. Eg MHC-PMS drug allergies req. • If drug allergy → if prescribed → warning to system user • If warning ignored → state reason.</p> <p>Related reqs. tests:</p> <ul style="list-style-type: none"> • Patient record - no allergy. Prescribe common allergy med. Check warning should not occur. • Patient record - known allergy. Prescribe allergy drug. Check warning. • Patient record - allergy to 2 or more drugs. Prescribe both drugs separately. Correct warning for each separately (one by one) • Prescribe 2 drugs with allergy. Check 2 warnings. • Issue drug with was allergy. Warning. Overrule warn. System should require to give reason. 	[5]	CO5	L3
OR				
2 (a)	<p>Explain various interface types and interface errors. Interface types (4M)</p> <p><u>Interface Types</u></p> <p>(i) Parameter interfaces - Data passed from one method or procedure to another.</p>	[8]	CO5	L2

- (ii) Shared memory interfaces - Block of memory is shared between procedures or functions.
- (iii) Procedural interfaces - One component encapsulates a set of procedures or functions to be called by other sub-systems. Objects and reusable components have this form of interface.
- (iv) Message passing interfaces - One component requests a service from another component by passing a message to it. e.g. client-server systems.

Interface Errors (4M)

Interface Errors

- (i) Interface misuse - A calling component calls another component and makes an error in its use of its interface. e.g. - wrong number or order of parameters.
- (ii) Interface misunderstanding - A calling component embeds assumptions about the behaviour of the called component which are incorrect. e.g. - searching binary search routine called with an unordered array.
- (iii) Timing errors - The called and calling component operate at different speeds and out-of-date information is accessed.

(b) Differentiate between verification and validation.

[2]

Verification (1M) Validation (1M)

Validation: Are we building the right product?

To ensure that software system meets the customer's expectations.

Verification: Are we building the product right?

To ensure that software conforms to its specification and meets its stated functional and non-functional requirements.

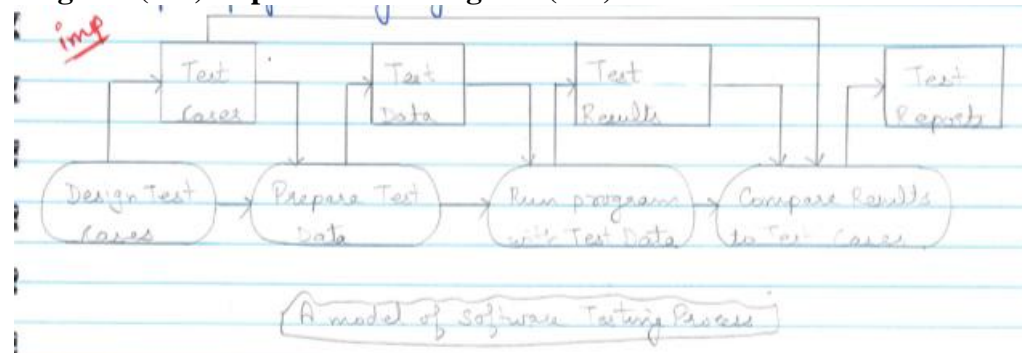
CO5

L1

PART B

3 (a) Explain testing process with the help of a neat diagram
Diagram (6M) Explanation of diagram (2M)

[8]



(b) Define test-driven development.
Definition (2M)

[2]

CO5

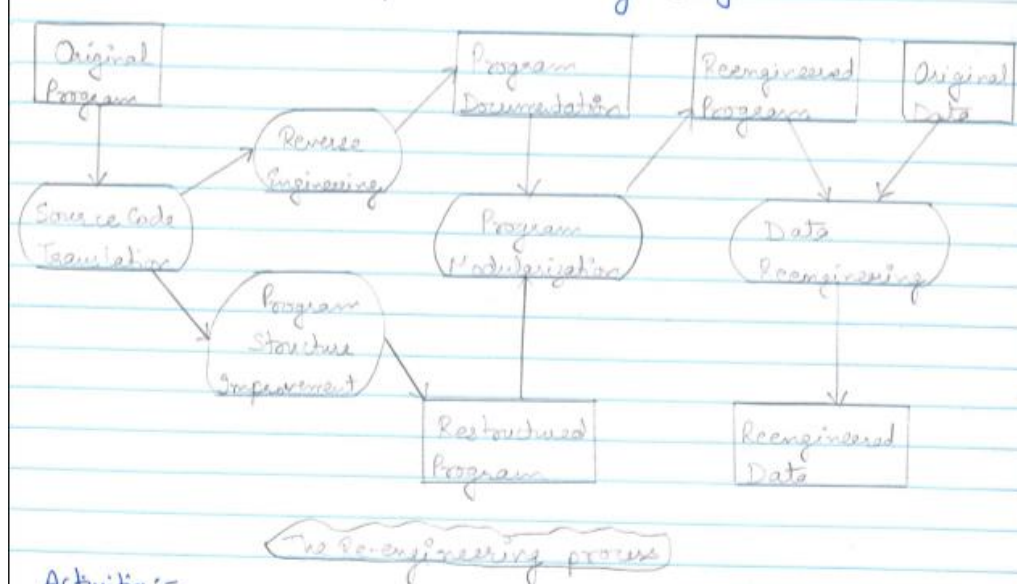
L1

- Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development.
- Tests are written before code and 'passing' the tests is the critical driver of development.
- You develop code incrementally, along with a test for that increment. you don't move on to the next increment until the code that you have developed passes its test.

OR

4 (a) Explain software reengineering with the help of a neat diagram. [6]

Software Reengineering diagram (6M)



(b) Explain maintenance cost factors that result in high cost of maintenance. [4]

Maintenance cost factors (4x1M)

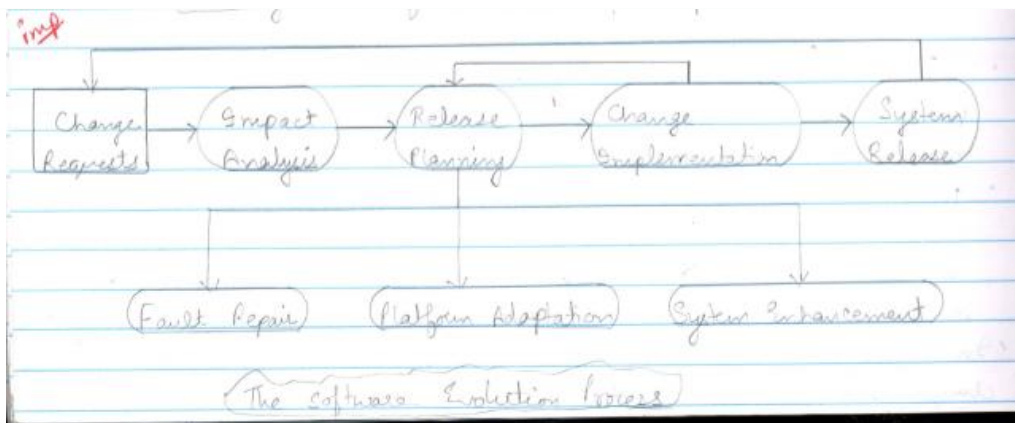
- Maintenance cost factors
1. Team stability - Development team is broken after product is delivered to work on new projects. It becomes difficult for maintenance team to understand system which takes lots of time & effort. If both teams are same, cost will be less.
 2. Contractual responsibility - Maintenance contract is different from development contract and may be given to a different organisations. So there is no incentive for development team to write software which is easy to change.
 3. Staff skills - Maintenance staff are often inexperienced and have limited domain knowledge. Old systems may be written in obsolete language.
 4. Program age and structure - As programs age, their structure is degraded and they become harder to understand and change.

PART C

5 (a) With the help of a neat diagram, explain software evolution process. [7]

Diagram (5M) Explanation (2M)

	CO4	L2
	CO5	L2
	CO5	L2



The cost and impact of proposed changes are assessed. If the proposed changes are accepted, a new release of the system is planned. During release planning, all proposed changes (fault repair, adaptation, and new functionality) are considered. A decision is then made on which changes to implement in the next version of the system. The changes are implemented and validated and a new version of the system is released. The process then iterates with a new set of changes proposed for the next release.

(b) Define three types of maintenance.
Types of maintenance (3x1M)

[3]

CO5 L2

Types of Maintenance

1. Corrective - Maintenance for fault repair.
Coding errors are relatively cheap to correct, design errors are more expensive and requirements errors are the most expensive to correct.
2. Adaptive - Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation.
3. Perfective - Maintenance to add or modify the system's functionality. The scale of changes required to the software is often much greater than for other types of maintenance.

OR

6 (a) What is program evolution dynamics? Explain Lehman laws.

[2+8]

CO5 L2

Program evolution dynamics definition (2M) Lehman laws (8x1M)

Program evolution dynamics is the study of the processes of system change.

Lehman's Laws

① Continuing change

A program that is used in a real-world environment must necessarily change, or else become progressively less useful in that environment.

② Increasing complexity

As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to ~~per~~ preserve and simplify the structure.

③ Large program evolution

Once system exceeds minimal size, it becomes more complex, hard to understand and leading to more likely for programmers to make errors. A large change may introduce more new faults

than the usefulness of the change to be delivered. Therefore, program evolution is a self-regulating process. System attributes such as size, time between releases, and number of reported errors is approximately invariant for each system release.

④ Organisational stability

Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development. Eg. communication overheads may dominate the work of team.

⑤ Conservation of familiarity

Over the lifetime of a system, the incremental change in each release is approximately constant as adding new functionality may introduce further faults.

⑥ Continuing growth

The functionality offered by systems has to continually increase to maintain user satisfaction.

⑦ Declining quality

The quality of systems will decline unless they are modified to reflect changes in their operational environment.

⑧ Feedback system

Evolution processes incorporate multi-agent, multi-loop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.

PART D

7 (a) Differentiate between milestones and deliverables.

[2]

CO5

L1

Milestones (1M) Deliverables (1M)

Milestones are points in the schedule against which you can assess progress, for example handover of the system for testing.

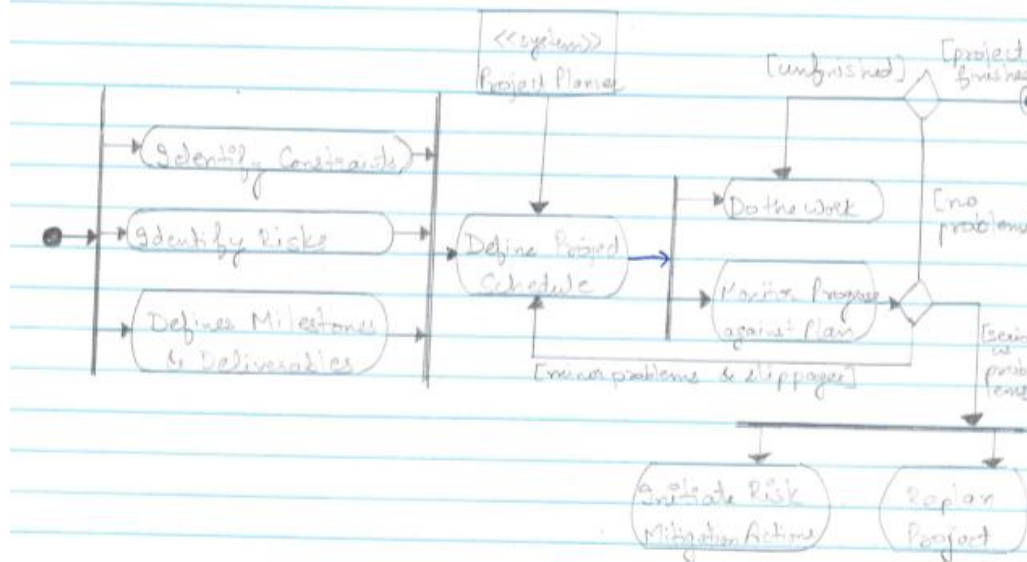
Deliverables are work products that are delivered to the customer eg., a requirements document for the system.

(b) Draw a neat diagram to explain project planning process.

[8]

CO4 L2

Diagram (6M) Explanation (2M)



Activity Diagram: The project planning process

- Assess constraints (required delivery date, staff available, overall budget, available tools etc), identify risks, milestones and deliverables.
- Draw up estimated schedule and start activities.
- Monitor progress against plan and make modifications to project schedule based on minor problems and slippage.
- If no problems arise, keep doing work & monitoring progress until project is finished.
- If some serious problems arise, initiate the risk mitigation actions and replan the project.

(c) Explain COCOMO-II model with a neat diagram

[10]

CO4 L2

(Diagram 6M) Explanation (4M)

- This is an empirical model that was derived by collecting data from a large number of software projects. The data were analyzed to discover the formulae that were the best fit to the observation.
- It is well-documented and non-proprietary estimation model and takes into account ^{more} modern approaches to software development.

Number of Application Points	Based on	Application Composition Model	Used for	Systems developed using Dynamic Languages, DB Programming etc.
Number of Function Points	Based on	Early Design Model	Used for	Initial effort estimation based on system requirements & design options
Number of LOC Reused or Generated	Based on	Reuse Model	Used for	Effort to integrate reusable components or automatically generated code
Number of Lines of Source Code	Based on	Post Architecture Model	Used for	Development effort based on System design specification

① Application Composition Model -

SW composed of existing parts, reuse

Effort estimated from size \rightarrow size estimated from Application points (reports, screens, modules of DB, scripting code etc.)

$$PM = \frac{NAP \times \left(1 - \frac{\% \text{ reuse}}{100}\right)}{PROD}$$

PM = Effort in Person-months

NAP - no. of App Points

PROD - Productivity

② Early Design Model -

- used when requirements are available but design has not started.

$$PM = A \times \text{Size}^B \times M$$

$$= 2.94 \times \text{Size}^{(1.1-1.24)} \times M$$

③ Reuse Model

- used to compute effort of integrating reusable components.

Black-box reuse - code not modified (generated)

White-box reuse - code modified (understood & integrated)

④ Post-architecture level -

- when system architecture is designed,

$$PM = A \times \text{Size}^B \times M$$

↳ 17 multipliers instead of 7.

OR

8 From the information given in the table, draw

(a) Bar chart/Gantt chart

[10]

CO4 L3

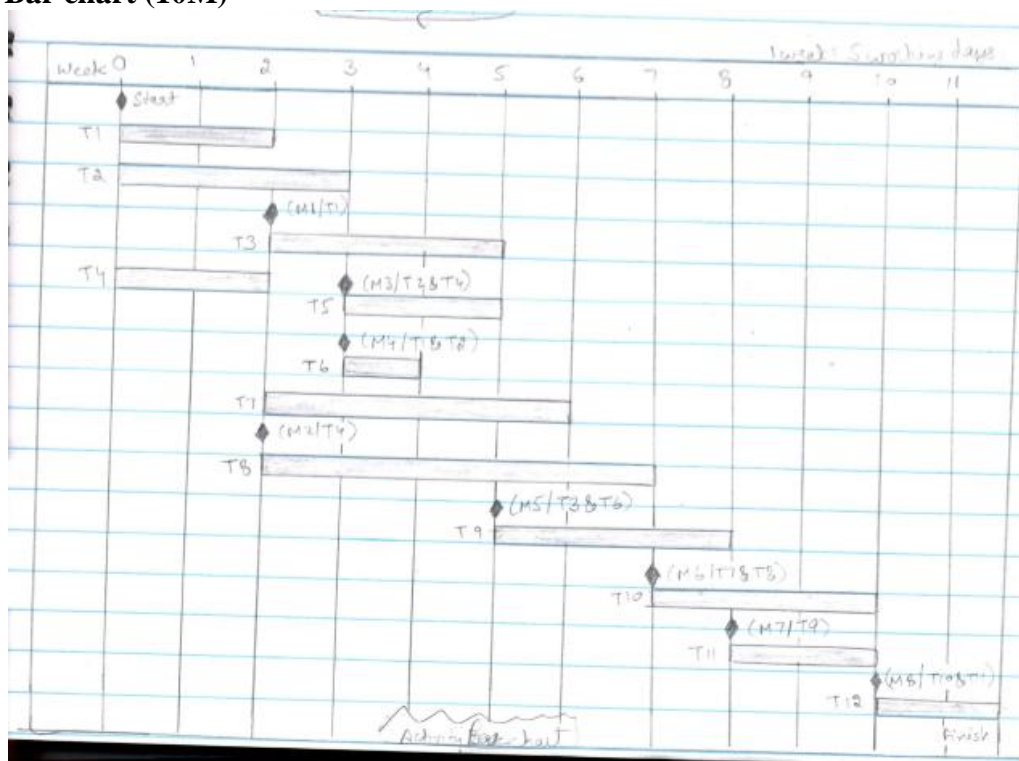
(b) Staff allocation chart

[10]

CO4 L3

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	-
T2	8	15	-
T3	20	15	T1 (M1)
T4	5	10	-
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

Bar chart (10M)

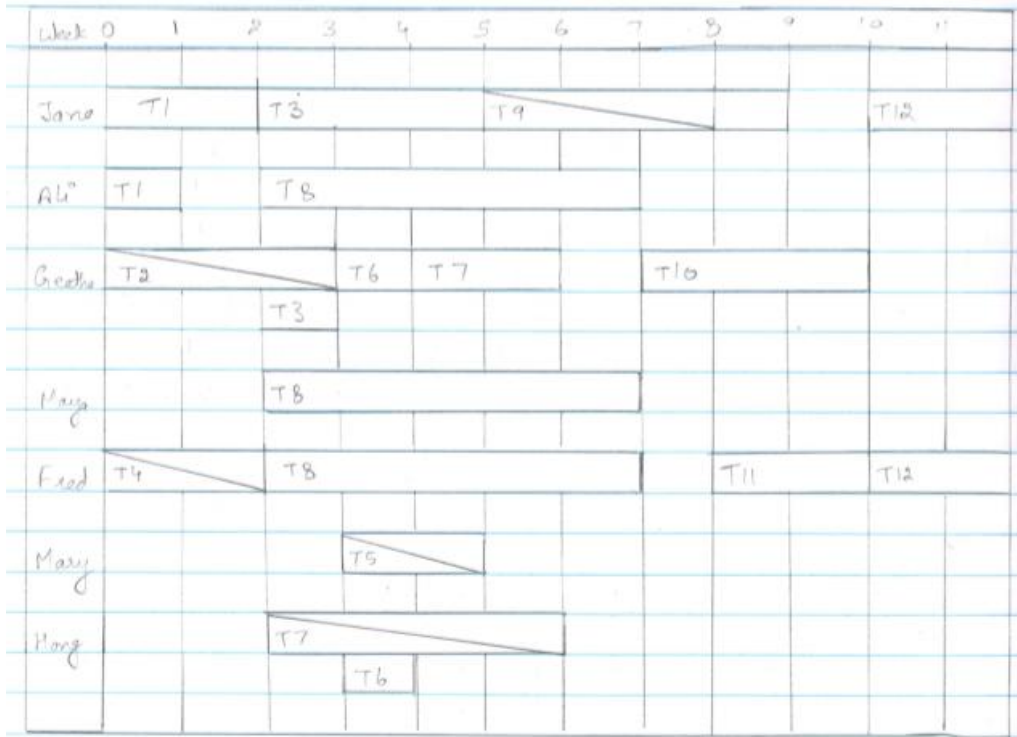


Staff allocation chart (10M) – It may be different for different students based on subjective selection

10

1 week = 5 working days

1 = part time



Staff Allocation Chart