

Section A

1. -Standalone applications. These are applications that are self-sufficient on a local computer and do not need any networking
-Embedded control applications. These are control applications that control and manage other hardware devices.
-Entertainment applications. These applications provide personal entertainment to users, such as games, music and videos.
-Modelling and simulation systems. These applications allow users to simulate and interact with virtual objects.
-Batch processing systems. These applications enable the processing of large batches of data.

2. a) Extreme programming is a software development approach where developers aim to complete as many requirements as they can in the current incremental phase.
b) Pair programming is a software development approach where developer pairs work together to develop software. Roles in pair programming involves the driver and the observer. The observer guides and helps the driver to spot overlooked coding problems or mistakes, while the driver does the actual work of writing code.

3. A) A user story is an agile artifact that documents detailed client requirements and includes the objective, priority and allocation for the task

b) SMART goals:

S – Specific. User stories should be specific in terms of target user, functionality and aim without leaving any ambiguity.

M – Measurable. User stories should be testable given a set of inputs and expected outputs.

Achievable

A – Attainable. User stories should be able to be developed within current developer technical capabilities.

R – Relevant. User stories should be relevant in the improvement or refactoring of the system product.

T – Time-boxed. User stories should be able to be developed within the limited time frame given by clients.

4. A) No. of members = 4

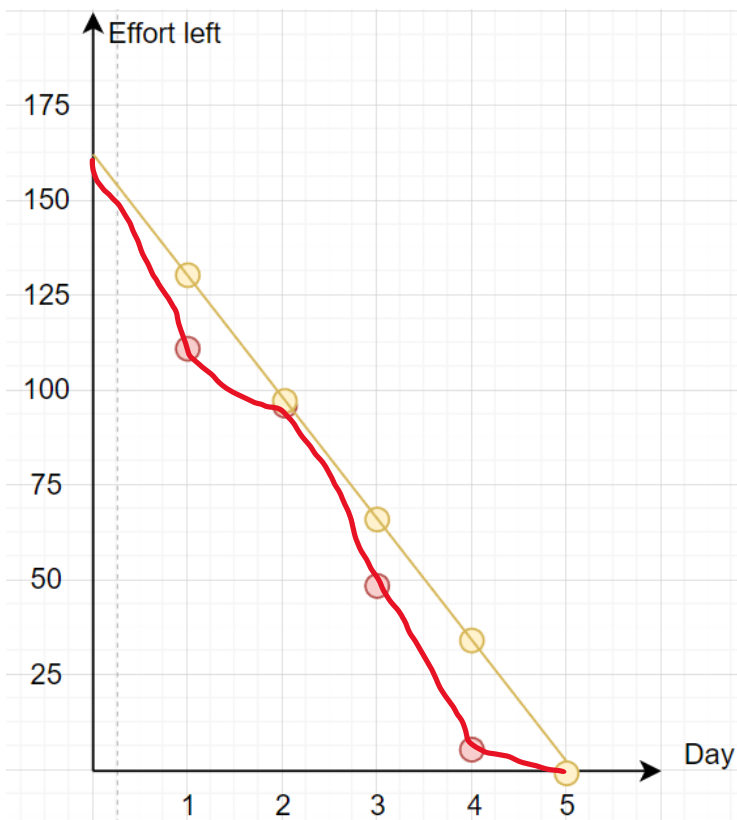
Working days = 5

Working hours per day = 8

Total effort = $5 \times 8 \times 4 = 160$ hours

Expected effort per day = $4 \times 8 = 32$ hours

Day	Expected effort	Actual effort	Expected burn	Actual burn
1	32	50	128	110
2	32	16	96	94
3	32	48	64	46
4	32	40	32	6
5	32	28	0	0



Expected – yellow

Actual – red

b) Yes, the team can accept the additional story. This is because the team has completed the sprint earlier than expected and actual burn of the sprint exceeds the effort required for the sprint. Therefore, the team can

expend the additional effort left over from the completed sprint to develop the additional story.

Section B

1. Application point count: $6 \times 1 + 2 \times 5 = 6 + 10 = 16$

New application point: $16 * 75\% = 12$

Effort to develop (person/month) = $12 / 13$

= 0.923 person/month

2. -Missing indentation. Indentation should be used to show different scopes of code. For example:

```
class employee{  
    private int employeeid;  
    private String name, department;  
    ...  
}
```

-Missing whitespace. Whitespace should be used to show different sections of related code to increase readability. For example.

```
class employee{  
    private int employeeid;  
    private String name, department;  
  
    public employee(){  
        ...  
    }  
    ...  
}
```

-Inconsistent naming convention. Variable names should strictly follow the chosen naming convention, such as delimited_case or camelCase, and class names should use PascalCase. For example:

```
class Employee{  
    private int employeeId;  
    private String name, department;  
    ...  
}
```


-Increase cohesion of setter. Parameters of the setEmployee() setter should be given their individual setters to decrease coupling of setters. For example:

```
public void setId(int id){  
    ...  
}  
  
public void setName(String name){  
    ...  
}
```


```
public void setDepartment(String department){  
    ...  
}
```


-separate class attributes into different lines. Class attributes should be separated into different lines to increase readability. For example:

```
class Employee{  
    private int employeeId;  
    private String name;  
    private String department;  
    ...  
}
```

3. A) Refactoring is a process of improving the performance or quality of the software without affecting the original functionality of the system.  **quality of code.**

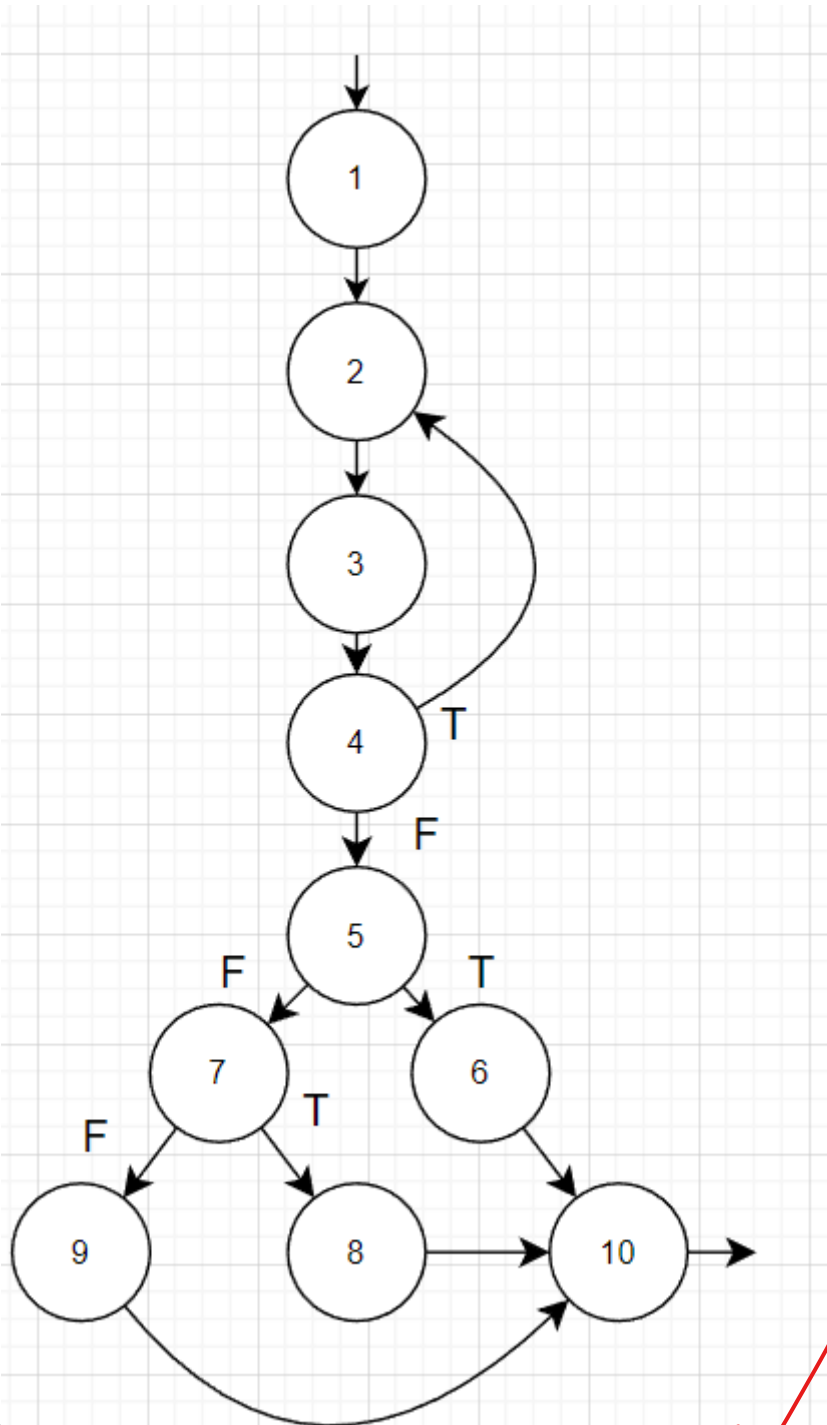
b) -Increase code quality. System code will seem more elegant and simple to the developer which increases readability and reduces developer confusion and frustration.

-Increase understandability of code. Refactored code will also make code sections to be  more understandable at initial glance, even for new developers.

-Increase maintainability of code. Refactored code allows the developer to easily perform  modifications to system settings when there is a policy change.

help in finding bugs too

Section C



1. A.

B. Cyclomatic complexity : 4

Independent paths:

[1,2,3,4,5,6,10]

[1,2,3,4,5,7,8,10]

[1,2,3,4,5,7,9,10]

[1,[2,3,4],5,6,10]



2. Availability is the amount of time the service or product is available to be used or requested by the user. Reliability is the capability of service or product to be able to fulfill user requirements without causing any system faults and affecting user processes.
3. A) Rate of fault occurrence. This is because hotel room reservation system manages a large number of reservations, and it is important to ensure that the system does not make any mistakes when handling a large amount of records.
B) Availability. This is because student attendance reports are frequently required by school staff, therefore it must be ensured that the system is always available to use.

Accepted, but to be more precise, the hotel room reservation should be evaluated by Availability as it may customers may need to use for managing the reservation records, it should be unavailable for less than 30 minutes per month.

For Student attendance report generator, it could be ROCOF. As this is not a critical system, faults are unlikely to cause severe disruption. It shall be able tolerate for 1 fault/100 hours of use.