

## Exercise Submission 2

Thi Van Anh DUONG Student ID: 90023112

Diploma of Information Technology, Curtin College

ISEN1000: Introduction to Software Engineering

Coordinator: Khurram Hameed

31 May 2022

### Student Declaration of Originality

<input checked="" type="checkbox"/>	This assignment is my own original work, and no part has been copied from another student's work or source, except where it is clearly attributed.
<input checked="" type="checkbox"/>	All facts/ideas/claims are from academic sources are paraphrased and cited/referenced correctly.
<input checked="" type="checkbox"/>	I have not previously submitted this work in any form for THIS or for any other unit; or published it elsewhere before.
<input checked="" type="checkbox"/>	No part of this work has been written for me by another person.
<input checked="" type="checkbox"/>	I recognise that should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with Curtin College policy.

### Electronic Submission:

<input checked="" type="checkbox"/>	I accept that it is my responsibility to check that the submitted file is the correct file, is readable and has not been corrupted.
<input checked="" type="checkbox"/>	I acknowledge that a submitted file that is unable to be read cannot be marked and will be treated as a non-submission.
<input checked="" type="checkbox"/>	I hold a copy of this work if the original is damaged, and will retain a copy of the Turnitin receipt as evidence of submission.

## Question 1: Planning, Software Project Management and Version Control

A. List the missing dependencies in Table 1.

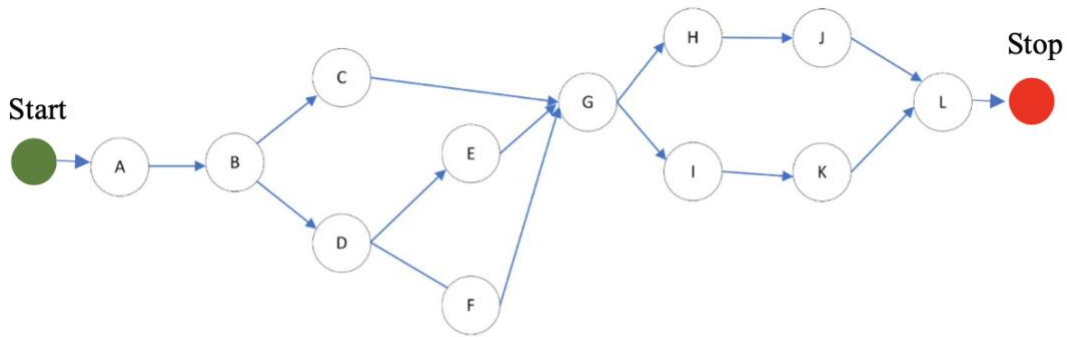


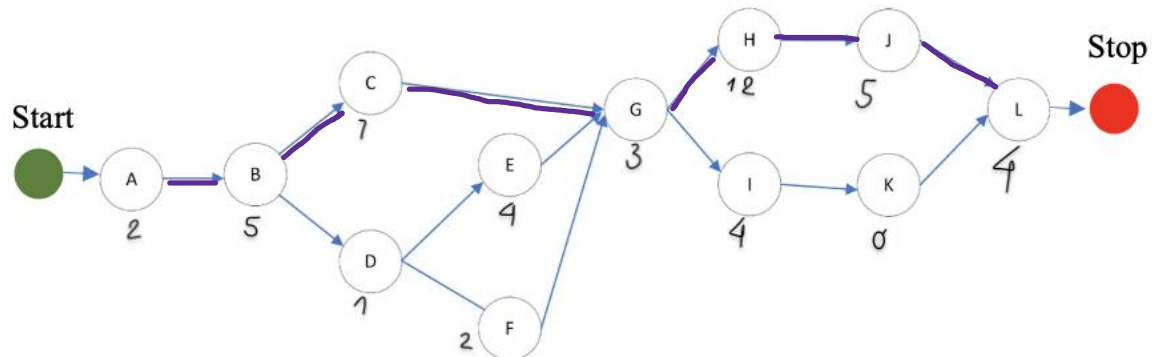
Fig. 1: 12 Nodes AON

Sr.	Task	Estimated Duration (Weeks)	Dependencies
1.	A	2	None
2.	B	5	A
3.	C	7	B
4.	D	1	B
5.	E	4	D
6.	F	2	D
7.	G	3	C, E, F
8.	H	12	G
9.	I	4	G
10.	J	5	H
11.	K	6	I
12.	L	4	K, J

B. Show your complete working on estimation of overall project duration, along with critical path estimation. (2 marks)

There are 6 estimated paths:

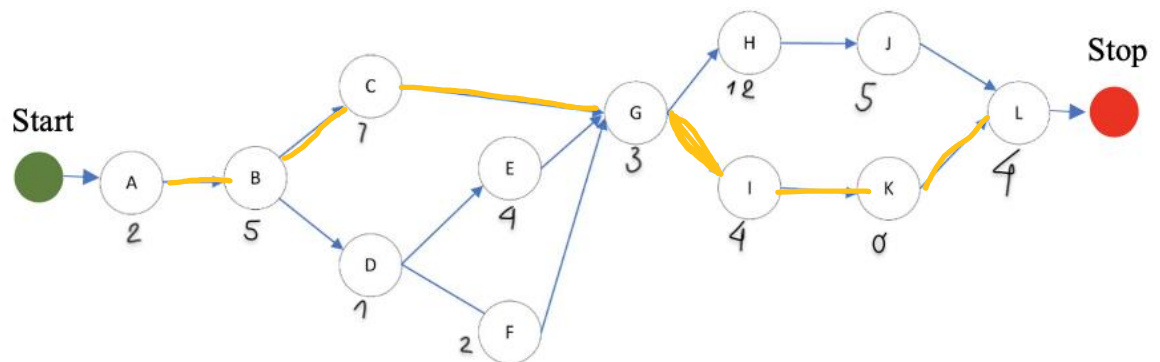
• **Path 1:**



Overall project duration:

$$A - B - C - G - H - J - L = 2 + 5 + 7 + 3 + 12 + 5 + 4 = \mathbf{38 \text{ Weeks}}$$

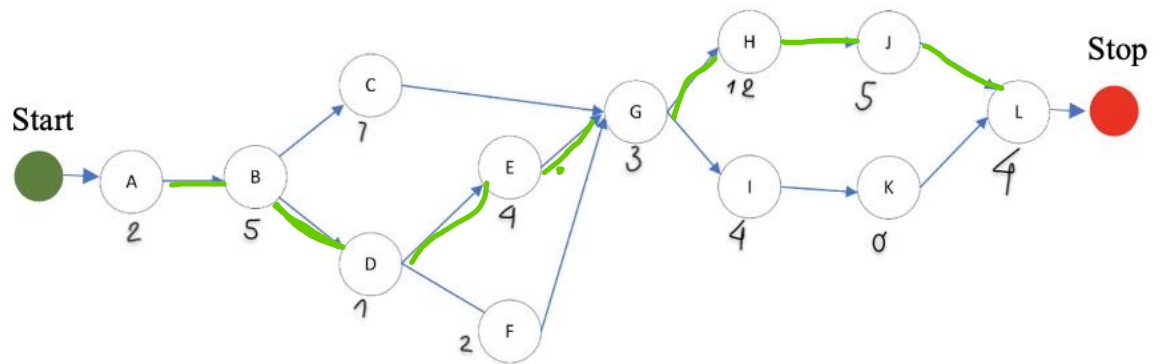
• **Path 2:**



Overall project duration:

$$A - B - C - G - I - K - L = 2 + 5 + 7 + 3 + 4 + 6 + 4 = \mathbf{31 \text{ Weeks}}$$

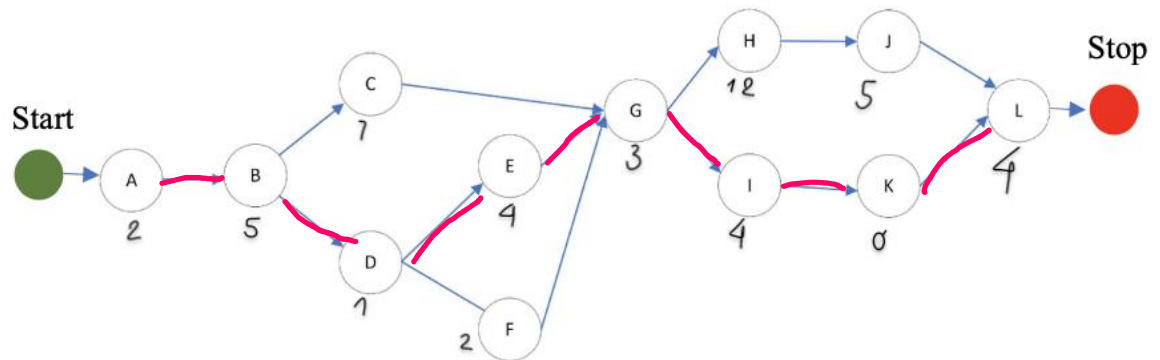
- **Path 3:**



Overall project duration:

$$A - B - D - E - G - H - J - L = 2 + 5 + 1 + 4 + 3 + 12 + 5 + 4 = \mathbf{36 \text{ Weeks}}$$

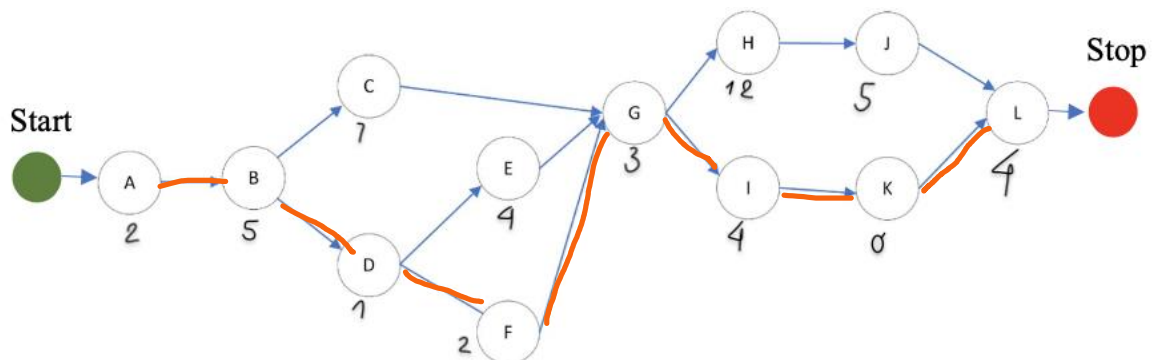
- **Path 4:**



Overall project duration:

$$A - B - D - E - G - I - K - L = 2 + 5 + 1 + 4 + 3 + 4 + 6 + 4 = \mathbf{29 \text{ Weeks}}$$

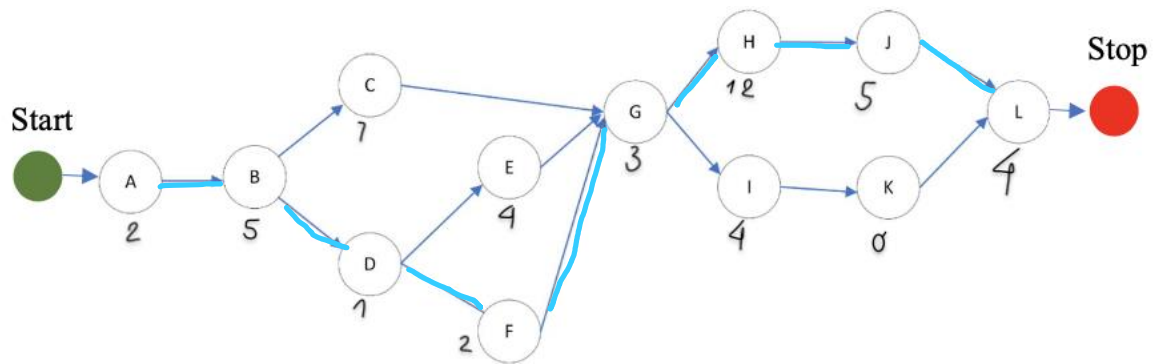
- **Path 5:**



Overall project duration:

$$A - B - D - F - G - I - K - L = 2 + 5 + 1 + 2 + 3 + 4 + 6 + 4 = \mathbf{27 \text{ Weeks}}$$

- **Path 6:**



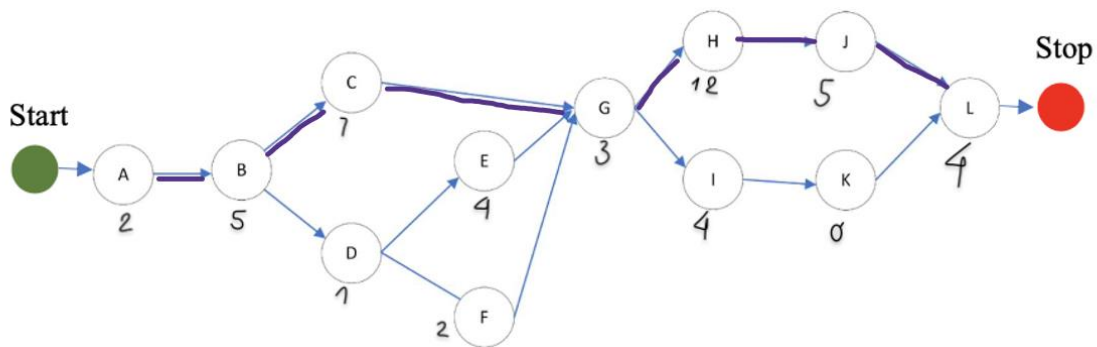
Overall project duration:

$$A - B - D - F - G - H - J - L = 2 + 5 + 1 + 2 + 3 + 12 + 5 + 4 = \mathbf{34 \text{ Weeks}}$$

$$\text{****Critical path} = \text{Max} \{ 38, 31, 36, 29, 27, 34 \} \\ = 38$$

Therefore, estimated critical path:

$$\mathbf{A \rightarrow B \rightarrow C \rightarrow G \rightarrow H \rightarrow J \rightarrow L}$$



C. Find out the task(s) which have slack time, and clearly mention the slack time of each identified task in tabular format. **(2 marks)**

D. Estimate the Early Start (ES), Late Start (LS), Early Finish (EF) and Late Finish (LF) for each task in the AON, present your answer in a tabular form. **(2 marks)**

- **Find the earliest start (ES) and earliest finish (EF) for each activity.**

- ⇒ Start with the activities that don't depend on anything (activity A)

- ⇒ Work forwards until get to the final activity (activity L).

- ⇒ Early Start Time = 0

- ⇒ **EF = ES + Duration**

- **Find the latest start (LS) and latest finish (LF) for each activity.**

- ⇒ Start with the activities at the end of the project (activity L)

- ⇒ Latest Finish of the activity L = Earliest Finish of the activity L

- ⇒ Work backward until get to the first activity.

- ⇒ **LS = LF – Duration**

- **Slack Time = LF – EF = LS – ES**

\*\*\* Activities that have slack time: D, E, F, I, K

\*\*\* Activities that do not have slack time (Slack time = 0): A, B, C, G, H, J, L

Then I have the table as below:

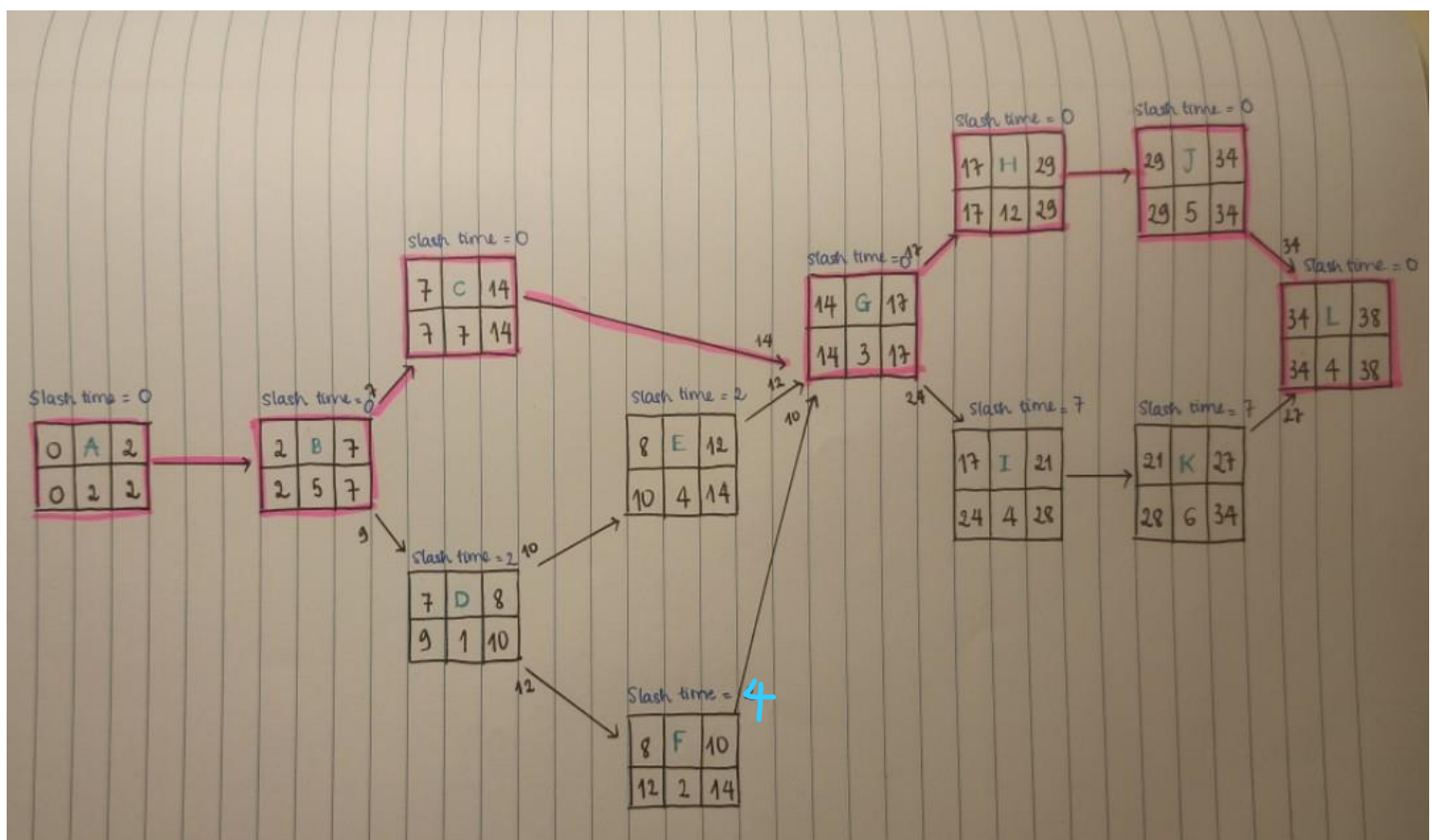
Activity	Duration	ES	EF	LF	LS	Slash Time
A	2	0	2	2	0	0
B	5	2	7	7	2	0
C	7	7	14	14	7	0
D	1	7	8	10	9	2
E	4	8	12	14	10	2
F	2	8	10	14	12	4
G	3	14	17	17	14	0
H	12	17	29	29	17	0
I	4	17	21	28	24	7
J	5	29	34	34	29	0
K	6	21	27	34	28	7
L	4	34	38	38	34	0

I use the diagram as below to calculate ES, EF, LS, LF:

ES	Activity	EF
LS	Duration	LF

$EF = ES + \text{Duration}$   
 $LS = LF - \text{Duration}$

Critical path is marked by pink color:





- E. You project start at week 0 and you have a sprint in 2 weeks, where project owner wants to add three activities (say C, D, E) as backlog, what necessary action would you consider in this case. **(4 marks)**

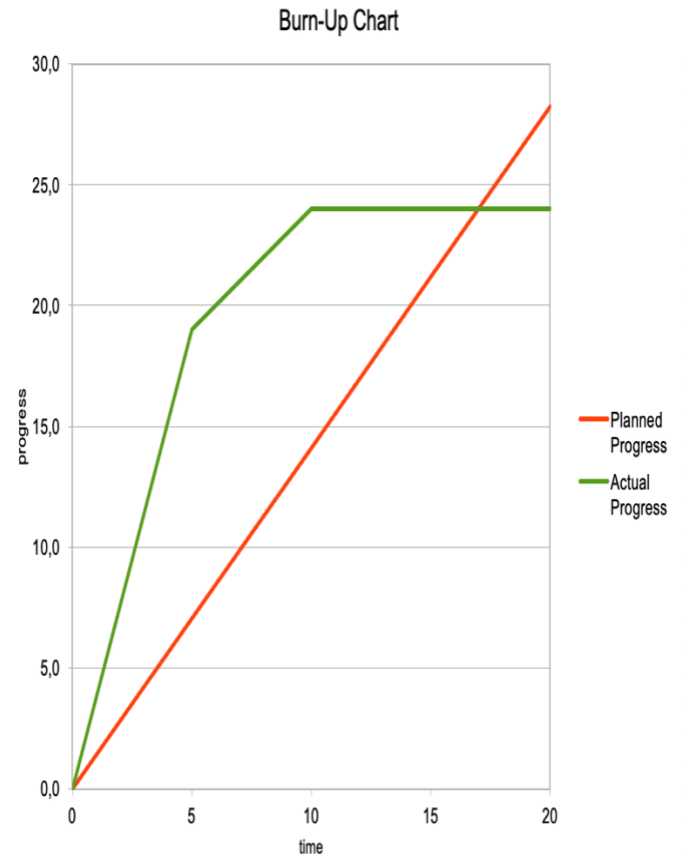
There are several actions I would consider:

- Having an official meeting with the development team to know whether or not they agree to add new activities at week 2 of the project (because they are the one who know the project properly).
- If the team agree, then discussing with the them about the 3 new activities the project owner wants to add: their dependencies, how long does it takes to complete (estimate)?, how urgent they are?, how many people will work with it? and make sure that the changes would not endanger to Sprint Goal and its quality.
- Ask project manager for necessary prototyping.
- Discuss with the team about the previous activities which are being worked (Can they skip some to prioritize new tasks, or they have to finish them first before doing other new tasks)
- Check whether or not the team can finish new tasks in current sprint, if not put it in waiting list and assign it in the next sprint.
- Adding more user stories in a running sprint.

- F. You take has completed activity “G” and all dependencies of “G” by the week 15. However, all other activities are not complete. Estimate actual performance on the Burn-up Chart. **(4 marks)**

I assume the actual finish time as the table below:

Activity	Estimated Duration (WEEKS)	Actual Finish	Time (actual weeks)	Planned Progress	Actual Progress
A	2	2	0	0,0	0
B	5	6	5	7,1	19
C	7	4	10	14,1	24
D	1	2	15	21,2	24
E	4	3	20	28,2	24
F	2	3			
G	3	4			
Sum of Task Durations	24				
Planned Project Duration	17	(Earliest Finish is 17)			
Planned Progress per Day	1,4				



G. Show a Kanban Board when the both

1. E and F are done/completed.

Todo	Coding/Testing	Review	Awaiting Merge	Done
G	C			A
H				B
I				D
J				E
K				F
L				

2. When activity “H” is under review.

Todo	Coding/Testing	Review	Awaiting Merge	Done
L	J	H		A
				B
				C
				D
				E
				F
				G
				I
				K

## Question 2: Functional and Non-functional requirements

### 1. Online examination system

A. Identify at least three actors for each application (human/non-human).

- Administrator (Human)
- Student (Human)
- Teacher (Human)
- Database (Non-Human)

B. Write one user story for each identified actor. The user stories should be significantly different from each other. **(3 marks)**

- As an administrator, I want to log into the system so that I can manage the exam information.
- As a student, I want to check my exam schedules so that I will not miss any important exam.
- As a student, I want to take online exams so that I can pass my unit.
- As a teacher, I want to add/edit/delete questions so that the exam will be more reliable.

C. Pick any two user stories from part A and give a full use case description for the selected user stories. The use cases must include 2 extensions for each use case. **(6 marks)**

#### ***Use case 1: Log in***

*Goal:* To log into the system.

*Primary actor:* **Administrator**

*Secondary actor:* Database

*Trigger:* Administrator click "log in" button to login to the system.

*Precondition:* administrator has an account, internet connection.

*Flow of event:*

1. Administrator selects 'Log in'
2. Login form appears.
3. Administrator types user name, password to the login form.
4. The system will double check user name, password of the administrator.
5. If the administrator types the wrong username and password the use case will move to extension 1A. Otherwise, continuing performing other features.
6. Use case ends.

*Extension:*

- 1A. Administrator login unsuccessfully:
  - I. The system announces that login process unsuccessfully.
  - II. Select "Sign in" or re-type user name and password. If selecting "Sign in", go to extension 1B.
  - III. The system asks member re-type user name and password.
  - IV. If the member agrees, the use case resumes at step 2 (FOE). Otherwise, the use case ends.
- 1B. Administrator does not have an account:
  - I. Sign in form appears.
  - II. Type necessary personal information to the form.
  - III. Click 'Submit' button.
  - IV. If sign in successfully, the system will back to log in page. The use case resumes at step 2 (FOE). If sign in unsuccessfully, the use case resumes at step 2 (Extension 1B)
  - V. The use case ends.

***Use case 2: Check exam schedule***

*Goal:* To check exam schedule.

*Primary actor:* **Student**

*Secondary actor:* Database

*Trigger:* Student click "View" option to check the upcoming exam schedules.

*Precondition:* student has an account, internet connection.

*Flow of event:*

1. Student log in Student Portal by Student Account.
2. Select the exam schedule option in the portal menu.
3. Student click "View" to observe their upcoming exam schedule.

*Extension:*

- 1A. Students forget their password:
  - I. The system asks student select "Forget password" option.
  - II. The system asks student to reset password via university email.
  - III. The use case resumes at step 1 (FOE)
- 1B. Exam schedule has not updated yet
  - I. The system will announce: Exam schedule is not available at the moment
  - II. The use case ends.

D. Identify which one would be the best reliability metric (MTTF, MTBF, ROCOF, POFOD or Availability) for each identified user story in part B. **(4 marks)**

- Mean Time To/Between Failure (MTTF/MTBF): The system must have a MTTF at least 1 year.
- Availability (AVAIL): The system should be available 99,999% of the time.
- Probability of Failure on Demand (POFOD): The system's configuration of questions and instant notifications feature should have a POFOD of 0.1% or less.
- Rate of Occurrence of Failure (ROCOF): The system must not have more than 1 question marked wrong is late in every 1000 answer sheets.

E. Identify 3 usability requirement for 3 user stories (select any 3, from part B) **(4 marks)**

- The exam system must allow students take online exam smoothly without any technical distractions.
- The exam system must allow student view their exam schedule in detail (date, time, location, time limit, etc.)
- The exam system must allow administrator update exam information and track examinees records (marks, attendance, etc.)
- The exam system must allow teacher post unit exam to student portal.

## 2. Flight auto pilot for airplane

A. Identify at least three actors for each application (human/non-human).

- Pilot (Human)
- Technical Engineer (Human)
- Database (Non-Human)
- Camera (Non-Human)
- GPS (Non-Human)
- Sensor (Non-Human)

B. Write one user story for each identified actor. The user stories should be significantly different from each other. **(3 marks)**

- As a pilot, I want to use auto navigation feature to control the plane automatically so that I can take a rest in a long journey.
- As a pilot, I want to enter a flight plan so that the plane can reach to right place.
- As a technical engineer, I want to check the connection of fuselage, wing, tail, tailplane, undercarriage and jet engine with autopilot system so that I can make sure the plane journey is safe.

- C. Pick any two user stories from part A and give a full use case description for the selected user stories. The use cases must include 2 extensions for each use case. (6 marks)

***Use case 1: Auto Flight***

*Goal:* To flight automatically.

*Primary actor:* **Pilot**

*Secondary actor:* Database, Camera, Sensor, GPS.

*Trigger:* Pilot click "Start" button to confirming auto-flight feature.

*Precondition:* turn on autopilot system, internet connection.

*Flow of event:*

1. Pilots check technical parameters of the plane.
2. Pilot enter flight details to the system.
3. Pilot click "Start" to switch into auto flight mode.

*Extension:*

- 1A. Lost connection with autopilot system:
  - I. Alarm announces to pilot nearby: "Lost connection!".
  - II. Switch into manual mode until signal is back.
  - III. The use case resumes at step 2 (FOE)
- 1B. Flight the wrong direction
  - I. The on-duty pilot detects it and turn off autopilot.
  - II. Pilot switches to manual mode.
  - III. The use case ends.

***Use case 3: Check exam schedule***

*Goal:* To record the connection between system and plane.

*Primary actor:* **Technical Engineer**

*Secondary actor:* Database, Sensor.

*Trigger:* Pilot click "Check" button to observe the connection.

*Precondition:* turn on autopilot system, internet connection.

*Flow of event:*

1. Turn on the system.
4. Enter the system menu and choose Settings.
5. Select "Connection".
6. Select "Check" to observe the system connection.

*Extension:*

- 1A. The connection is not stable:
  - I. Technical engineer checks the unstable part.
  - II. Report to engineer team and start fixing before flight journey.
  - III. Reset the system.
  - IV. The use case resumes at step 2 (FOE).

1B. The connection is good:

- I. Technical Engineer makes a report.
- II. The use case ends.

D. Identify which one would be the best reliability metric (MTTF, MTBF, ROCOF, POFOD or Availability) for each identified user story in part B. **(4 marks)**

- Mean Time To/ Between Failure (MTTF/MTBF): The system must have a MTTF of at least 2 years.
- Availability (AVAIL): The system should be available 99,999% of the time.
- Probability of Failure on Demand (POFOD): The system's auto navigation feature should have a POFOD of 0.1% or less.
- Rate of Occurrence of Failure (ROCOF): The system must not make more than 1 accident in every 1000 hours.

E. Identify 3 usability requirement for 3 user stories (select any 3, from part B) **(4 marks)**

- The system must allow pilot control the airplane's speed.
- The system must allow technical engineer calculate the amount of fuel needed for a flight journey.
- The system must maximize the security so that hacker cannot the data from system.



### Question 3: Unit Testing

A. Design a test case for the Equivalence Partitioning testing. (7 marks)

A function takes two string inputs (s1, s2), compares the lengths of both strings, take the lowest string length (n) out and return the characters of string "s1" equal to the lowest string length. If "s1" is smaller than "n" it returns empty string ("").

Category	Test Data	Actual Output
n is length of s1	s1 = coder s2 = programmer	coder
n is length of s2	s1 = different s2 = seafood	differe
Length s1 = length s2	s1 = Xinchao s2 = Xinchao	Xinchao
s1 has length smaller than n	s1 = "" s2 = a	""

```
def printString(s1, s2):  
    if len(s1) < len(s2):  
        n = len(s1)  
        return(s1[:n])  
    elif len(s2) < len(s1):  
        n = len(s2)  
        return(s1[:n])  
    elif len(s1) == len(s2):  
        n = len(s1)  
        return(s1[:n])  
    else:  
        n = len(s1)  
        return("")
```

```
s1 = input("Enter string s1: ")  
s2 = input("Enter string s2: ")  
print(printString(s1,s2))
```

Question 3\_A.py

**B. Design a test for the following for the Boundary Values Analysis (BVA). (5 marks)**

A function calculates the student concessions on the transport tickets based on the age category. It imports an age (integer), and ticket price (floating point number). For (ages from 1-5), the discount is 5%. For (ages from 6-16), the discount is 15%. There is no discount for (ages from 17- 35). For (ages 36-49) the discount is 30%. There is 50% discount for ages (50 and above) If age is invalid, the function returns zero (real).

Note:

- Invalid is Age < 0.
- Ticket fee is \$10 as default for traveling 2 zones.

Boundaries (Age)	Test data	Discount	Expected result
Invalid/1	Age = -3 Age = 1	0% 5%	0.0 9.5
5/6	Age = 5 Age = 6	5% 15%	9.5 8.5
6/17	Age = 16 Age = 17	15% 0%	8.5 10
35/36	Age = 35 Age = 36	0% 30%	10 7
49/50	Age = 49 Age = 50	30% 50%	7 5
50 and above	Age = 80	50%	5

```

def discount(age, ticketPrice):
    if age < 0:
        print("Invalid age!")
        price = 0.0
        return price
    elif age >= 1 and age <= 5:
        price = ticketPrice*(1-5/100)
        return price
    elif age >= 6 and age <= 16:
        price = ticketPrice*(1-15/100)
        return price
    elif age >= 17 and age <= 35:
        price = ticketPrice*(1-0/100)
        return price
    elif age >= 36 and age <= 49:
        price = ticketPrice*(1-30/100)
        return price
    elif age >= 50:
        price = ticketPrice*(1-50/100)
        return price

age = int(input("Enter age: "))
ticketPrice = float(input("Enter ticket price: "))
print(discount(age, ticketPrice))

```

Question 3\_B.py

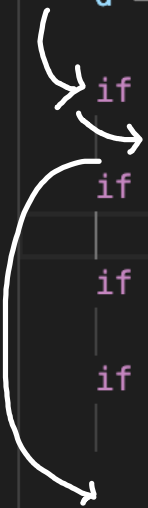
C. Consider the following python code to read a python code to find the largest of 4 numbers, develop a test design for white box testing. (7 marks)

- Identify the path through production code.
- Select test data for each test case.
- For each test case, determine the expected result.

➔ Number of path: 4

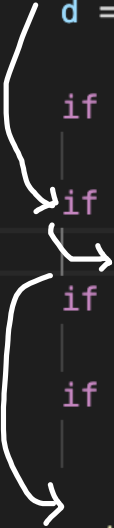
First path:

```
def max():  
    a = input("Enter 1st integer: ")  
    b = input("Enter 2nd integer: ")  
    c = input("Enter 3rd integer: ")  
    d = input("Enter 4th integer: ")  
  
    if a >= b and a >= c and a >= d:  
        result = a  
    if b >= a and b >= c and b >= d:  
        result = b  
    if c >= a and c >= b and c >= d:  
        result = c  
    if d >= a and d >= b and d >= c:  
        result = d  
  
    print(result)
```



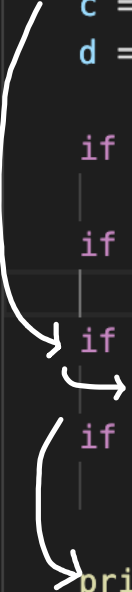
Second path:

```
def max():  
    a = input("Enter 1st integer: ")  
    b = input("Enter 2nd integer: ")  
    c = input("Enter 3rd integer: ")  
    d = input("Enter 4th integer: ")  
  
    if a >= b and a >= c and a >= d:  
        result = a  
    if b >= a and b >= c and b >= d:  
        result = b  
    if c >= a and c >= b and c >= d:  
        result = c  
    if d >= a and d >= b and d >= c:  
        result = d  
  
    print(result)
```



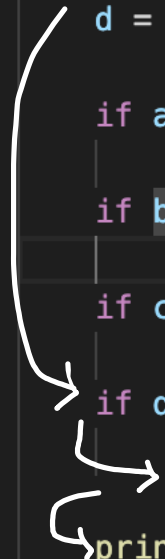
Third path:

```
def max():  
    a = input("Enter 1st integer: ")  
    b = input("Enter 2nd integer: ")  
    c = input("Enter 3rd integer: ")  
    d = input("Enter 4th integer: ")  
  
    if a >= b and a >= c and a >= d:  
        result = a  
    if b >= a and b >= c and b >= d:  
        result = b  
    if c >= a and c >= b and c >= d:  
        result = c  
    if d >= a and d >= b and d >= c:  
        result = d  
  
    print(result)
```



Fourth path:

```
def max():  
    a = input("Enter 1st integer: ")  
    b = input("Enter 2nd integer: ")  
    c = input("Enter 3rd integer: ")  
    d = input("Enter 4th integer: ")  
  
    if a >= b and a >= c and a >= d:  
        result = a  
    if b >= a and b >= c and b >= d:  
        result = b  
    if c >= a and c >= b and c >= d:  
        result = c  
    if d >= a and d >= b and d >= c:  
        result = d  
    print(result)
```



Path	Test Data	Expected result
Enter first if statement	a = 9 b = 7 c = 7 d = 2	9
Enter second if statement	a = 5 b = 8 c = 5 d = 6	8
Enter third if statement	a = 7 b = 1 c = 21 d = 13	21
Enter fourth if statement	a = 1 b = 2 c = 3 d = 4	4

**D. Implement your test design in parts A, B and C using python (unittest) modules. (6 marks)**

My test case:

```
def test_discount(self):
    self.assertEqual(0.0, Question3B.discount(-3,10), "Invalid age!")
    self.assertEqual(9.5, Question3B.discount(1,10), "5% discount")

    self.assertEqual(9.5, Question3B.discount(5,10), "5% discount!")
    self.assertEqual(8.5, Question3B.discount(6,10), "15% discount")

    self.assertEqual(8.5, Question3B.discount(16,10), "15% discount")
    self.assertEqual(10, Question3B.discount(17,10), "0% discount")

    self.assertEqual(10, Question3B.discount(35,10), "0% discount")
    self.assertEqual(7, Question3B.discount(36,10), "30% discount")

    self.assertEqual(7, Question3B.discount(49,10), "30% discount")
    self.assertEqual(5, Question3B.discount(50,10), "50% discount")

    self.assertEqual(5, Question3B.discount(80,10), "50% discount")

import unittest
import sys, io
import Question3A
import Question3B
import Question3C

class TestSuite(unittest.TestCase):

    def test_printString(self):
        capOut = io.StringIO()
        sys.stdout = capOut
        Question3A.printString("coder","programmer")
        self.assertEqual("coder\n", capOut.getvalue(), "n is length of s1")

        capOut = io.StringIO()
        sys.stdout = capOut
        Question3A.printString("different","seafood")
        self.assertEqual("differe\n", capOut.getvalue(), "n is length of s2")

        capOut = io.StringIO()
        sys.stdout = capOut
        Question3A.printString("Xinchao","Xinchao")
        self.assertEqual("Xinchao\n", capOut.getvalue(), "n = length s1 = length s2")

        capOut = io.StringIO()
        sys.stdout = capOut
        Question3A.printString("", "a")
        self.assertEqual("\n", capOut.getvalue(), "s1 has length smaller than n")

    def test_maxNumber(self):
        capOut = io.StringIO()
        sys.stdout = capOut
        sys.stdin = io.StringIO("9\n7\n8\n2")
        Question3C.maxNumber()
        self.assertEqual("Enter 1st integer: Enter 2nd integer: Enter 3rd integer: Enter 4th integer: 9\n", capOut.getvalue(), "a is largest number")

        capOut = io.StringIO()
        sys.stdout = capOut
        sys.stdin = io.StringIO("5\n8\n5\n6")
        Question3C.maxNumber()
        self.assertEqual("Enter 1st integer: Enter 2nd integer: Enter 3rd integer: Enter 4th integer: 8\n", capOut.getvalue(), "b is largest number")

        capOut = io.StringIO()
        sys.stdout = capOut
        sys.stdin = io.StringIO("7\n1\n21\n13")
        Question3C.maxNumber()
        self.assertEqual("Enter 1st integer: Enter 2nd integer: Enter 3rd integer: Enter 4th integer: 21\n", capOut.getvalue(), "c is largest number")

        capOut = io.StringIO()
        sys.stdout = capOut
        sys.stdin = io.StringIO("1\n2\n3\n4")
        Question3C.maxNumber()
        self.assertEqual("Enter 1st integer: Enter 2nd integer: Enter 3rd integer: Enter 4th integer: 4\n", capOut.getvalue(), "a is largest number")
```

Run python3 -m unittest Quesyion3D:

```
ccadmin@CCUbuntu64bit:~/ISEN1000/Assignment2$ python3 -m unittest Question3D
Invalid age!
...
-----
Ran 3 tests in 0.000s

OK
```



## Question 4: Modularity

- A. You are restricted to communicate through to the global variable(s) only among the functions listed. Show how this can be performed. (5 marks)

```
# A function reads marks of a student for different units from a file/record store them in an array
# Assume I already had marks.csv (is the file contains student's marks)
def readMarks():
    fileobj = open('marks.csv', 'r')
    data = fileobj.readlines()
    while data:
        mark = data.strip().split(',')
        marks.append(mark)
    fileobj.close()
    return marks

# A function counts the values of an array which are less than 50 and store the count in a variable
def counts(marks):
    count = 0
    for i in range (len(marks)):
        if marks[i] < 50:
            count += 1
    return count

# A function prints the prints an array's element
def prints(marks):
    print("List of marks: ")
    for i in range (len(marks)):
        print(marks[i])

if __name__ == "__main__":

# Global variables
    marks = []
    studentMarks = readMarks()
    countMarks = counts(studentMarks)
    # print a count variable - I don't want to put it in prints function because it will duplicate the code!
    print("Marks less than 50 is: ", countMarks)
    prints(studentMarks)
```

- **marks** is a global variable will be updated by the function **readMarks** from marks.csv, elements in that array will be compared with 50 to find out the element less than 50 and it will be display by the **counts** function.
- Because I am restricted to communicate through global variables, I created this variable by **readMarks** function and return it. My **counts** function will take **marks** array as an input parameter and produce **count**. Similarly, my **prints** function also use the same **marks** array to display the element of the array.

B. Demonstrate how the flag(s) can be used to combine the listed methods into one method. (5 marks)

```
def controlFlag():
    check = False # control flag variable

    fileobj = open('marks.csv', 'r')
    data = fileobj.readlines()
    while data:
        mark = data.strip().split(',')
        fileobj.close()

    if len(mark) > 0:
        check = True
    if check == True:
        count = 0
        i = 0
        while i < len(mark):
            studentMarks = int(mark[i])
            if studentMarks < 50:
                count += 1
            i += 1
        print("Student's marks: ")
        for i in range (len(mark)):
            print(mark[i])
        print("Marks < 50: ", count)
    else:
        print("Marks file is empty!")

if __name__ == "__main__":
    controlFlag()
```

- I use **check** variable as a control flag to combine my three functions.
- Initially, I assigned **check** = False as default. The program will continue read data from the file. According to my code above, if data is valid, **check** will be assigned to True. From there, I will have enough conditions to start my count and prints function.
- Thank to **check** variable my three functions are linked.

- C. The control flag(s) and global variable has significant modularity issue, demonstrate how both can be avoided yet maintain the functionality. **(5 marks)**

```
def avoidModule():

    fileobj = open('marks.csv', 'r')
    data = fileobj.readlines()
    while data:
        mark = data.strip().split(',')
        fileobj.close()

        if len(mark) > 0:
            count = 0
            i = 0
            while i < len(mark):
                studentMarks = int(mark[i])
                if studentMarks < 50:
                    count += 1
                i += 1
            print("Student's marks: ")
            for i in range (len(mark)):
                print(mark[i])
            print("Marks < 50: ", count)
        else:
            print("Marks file is empty!")

if __name__ == "__main__":
    avoidModule()
```

- In question B, I already removed the global variable (**marks**). In this part, in order to remove the control flag but still retain the functionality, I removed **check** variable and just simply check the length of max to make sure it has data in that, then continue doing counting and printing as above.

D. Discuss in detail the issues with techniques in part A and B, how these issues are resolved in part C. (5 marks)

**Issue in part A:** Global variable.

It causes coupling issues like:

- Tight coupling: **marks** array is greatly linked functions together. If when we take wrong data from the file and store it in **marks**, it will greatly affect **count** and **prints** function (wrong answer, semantic errors).
- Time-consuming to put it into function, test and debug.
- Increase complexity of the program. Specifically, when the developer itself or developer team look back the code, they will find it difficult to understand.

**Issue in part B:** Control flag

It causes:

- Coupling issue: if something wrong with **check** variable, the whole program will break (tight coupling) → difficult to test and debug.
- Cohesion issue.
- Create new **check** variable is unnecessary → waste computer RAM and make program running slower.

**Solution in part C:**

- Put everything into 1 function and use **mark** as a local variable to remove global variable → reduce coupling issue, make the program readable.
- Remove control flag variable (**check**) to avoid unnecessary variables → reduce coupling issue and increase cohesion.

## Question 5: Ethics and Professionalism

**Part A:** An online parcel tracking software (Functional requirement)

Online parcel tracking software was used by online commerce sellers to check their products status to customers.

When designing the system, the software team made a use case to view the parcel status:

*Extension:*

- 1A. Login unsuccessfully.
  - I. The system asks seller select “Forget password” option.
  - II. The system ask seller to reset password via email.
  - III. The system ask seller enter password.
  - IV. The use case resumes at step 1.

Due to time restriction and some inconvenience, the development team ignored the password conditions (like: password must have at least one capital character and special character). Therefore, hacker can easily steal user’s password, enter to user’s account to steal shipping tracking number or receiver’s address. These hackers will sell these information for scammers, who will rely on customer information, product information and shipping codes to swap the product for a cheap product or scrap rubbishes to trick customers.

As a result, customers when received unwanted products will be angry and stop using shop’s service, the shop also lost its reputation and huge amount of money in case they get sued.

## **Part B: Curtin College contact management software (Non-functional requirements).**

A software development team was hired to design Curtin College contact management software. This system will store personal information of students, unit coordinators and tutors: name, phone number, email address and home address. College's administration, teachers will use data in this system to send out notifications about classes, registration information or college events.

When design the software system, the development team had **Rate of Occurrence of Failure (ROCOF)**: The system must not report more than 1 incorrect personal details in every 1000 searches. However, when designing the system, the team ignored to update and replace the new database to the system when student or teacher change email address or phone number due to personal problems. Therefore, the old email addresses are still remained and not updated yet.

One day, unit coordinator make announcement about an assessment (This assessment costs 40% of the unit): "The class assessment will happen 1 days earlier compared to schedule on the Moodle". However, due to the negligence in updating the system, the unit coordination accidentally sends the aforementioned announcement to student's old email address. Apparently, students could not receive the important piece of information and face with the probability of failing the unit. It will certainly put them under pressure, take them surplus amount of time to make clarification with teacher and related department.

## **Part C: Online food ordering system at a restaurant (Project Management)**

A restaurant hires a software team to design an online food ordering system for them in order to reach new potential customers.

Initially, the development team, project manager and client had official meetings and agreed several features for the software: set up menu, ordering food, finding suitable food delivery brand, restaurant opening status, food stock status, driver's position. This project uses agile management method to track and control the whole project, each sprint lasts not longer than 4 weeks.

On the half way of the project, customers accidentally want to add new features for the system: discount features, finding nearest restaurant chains and new payment method and they want to it being added as soon as possible. The project manager wants to satisfy the customer so he agreed to add these features for customers before asking opinion of development team. He added these features in the happening sprint instead of in backlog. This decision makes each sprint overload (tasks in sprint up to 4 – but usually just 2-3). This action makes the whole team fell in chaos and put them under stress because they did not know what they should prioritize, related algorithms and limited time (4 weeks) to finish all task in one sprint.

As a result, due to the restricted amount of time and too much tasks need to finish, the team does not have enough time to review and debug. The three added tasks and the other tasks also were broken when put in practice, customer was angry and the company itself lost reputation.