


18+14+17+21+15 = 85/100
Excellent

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Formal Methods	Course Code:	SE2003
	Degree Program:	BS-SE	Semester:	Spring 2023
	Exam Duration:	3 Hours	Total Marks:	100
	Paper Date:	3rd June.2023	Weight	40%
	Section:	-	Page(s):	7
	Exam Type:	Final Exam	Instructor	Dr.Wafa Basit

Student : Name _____

Roll No. _____

Section: 6A

Instruction/Notes: Attempt all questions. Make necessary assumptions where required, Give justification for your answers. Don't use lead pencil. Solve on the question paper. Draw neat and understandable diagrams

Question # 1 (5+5+5+5+5)

a) What is the difference between an FSA and a Petrinet?

→ A petrinet is a State and transition diagram which stimulates wide range of use cases like Deadlocks, Synchronization through tokens.

→ Meanwhile, FSA also have states and transition but it cannot illustrate scenarios where there will be multiple users or arrays.

b) What is the difference between Explicit and Symbolic Model checking?

→ In explicit model checking state explosion occurs when number of states is very large. SPIN is an example.

→ In symbolic model checking this issue was addressed through a tool NuSMV. It makes the use of logical symbols on states.

c) What is a verification condition generator?

VC generator generates a logical formula through "compiling" the program in low level code and using specifications. Its formula is presumed to be logically valid which is verified by theorem prover.

d) Does refactoring change the externally observable behavior of code?

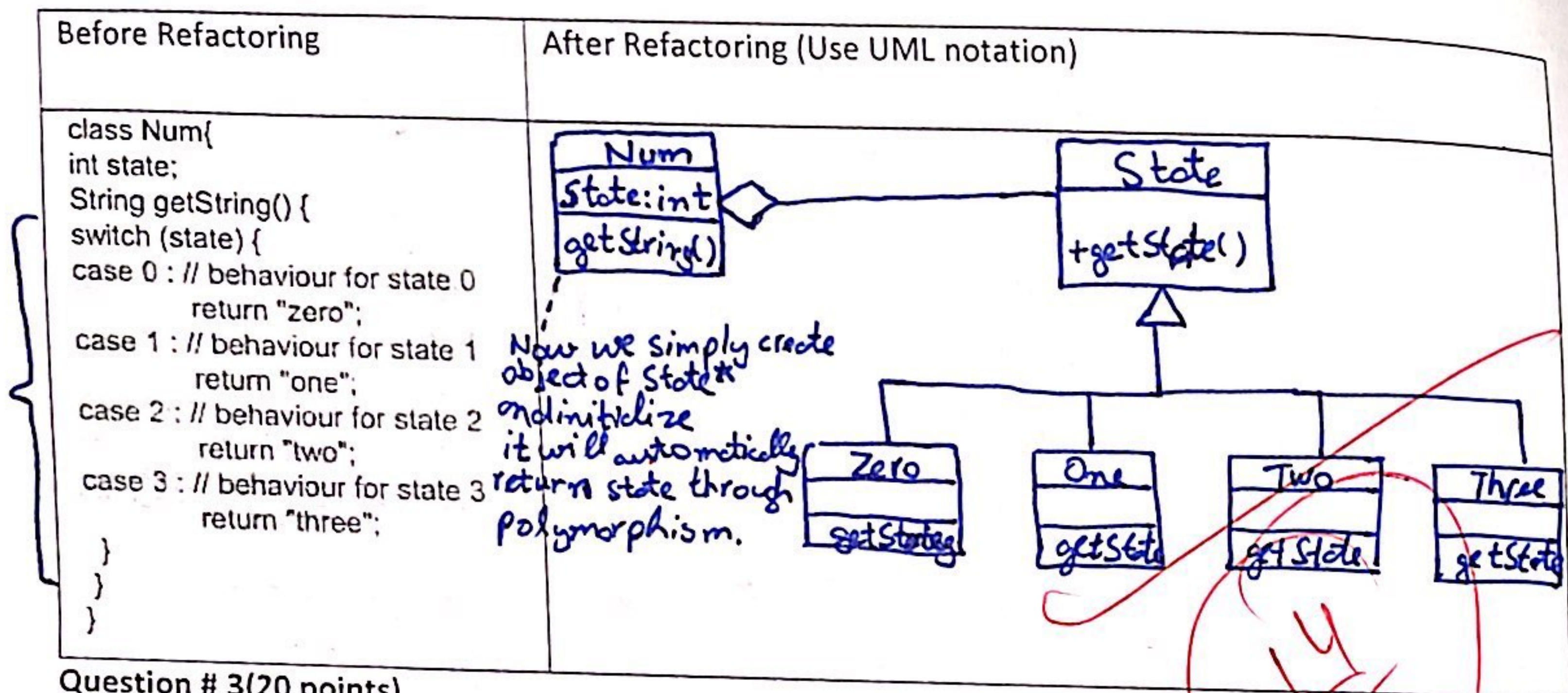
No, refactoring should be done in such a way that external behaviour remains same, although code gets cleaner and easy to understand.

e) What is the safety net for refactoring?

A safety net is a limit till where you can refactorize your code. If it is reached or broken then behaviour of code changes for user which is wrong.

Question # 2 Identify the bad smells in the following code. Also, mention and apply the suitable refactoring (15 points)

The whole switch case part is a bad smell. To fix it, we can use the type class state with Strategy Design Pattern

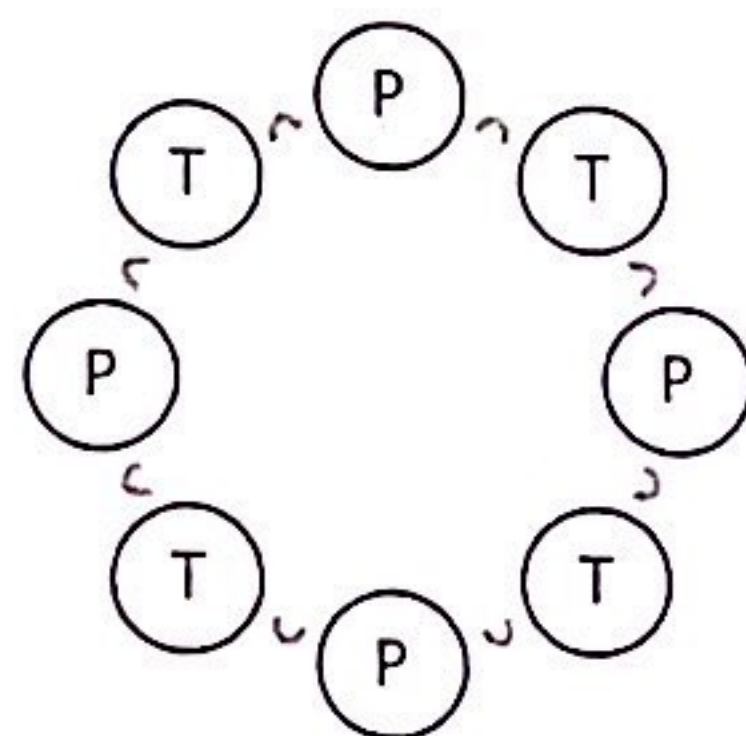


Question # 3 (20 points)

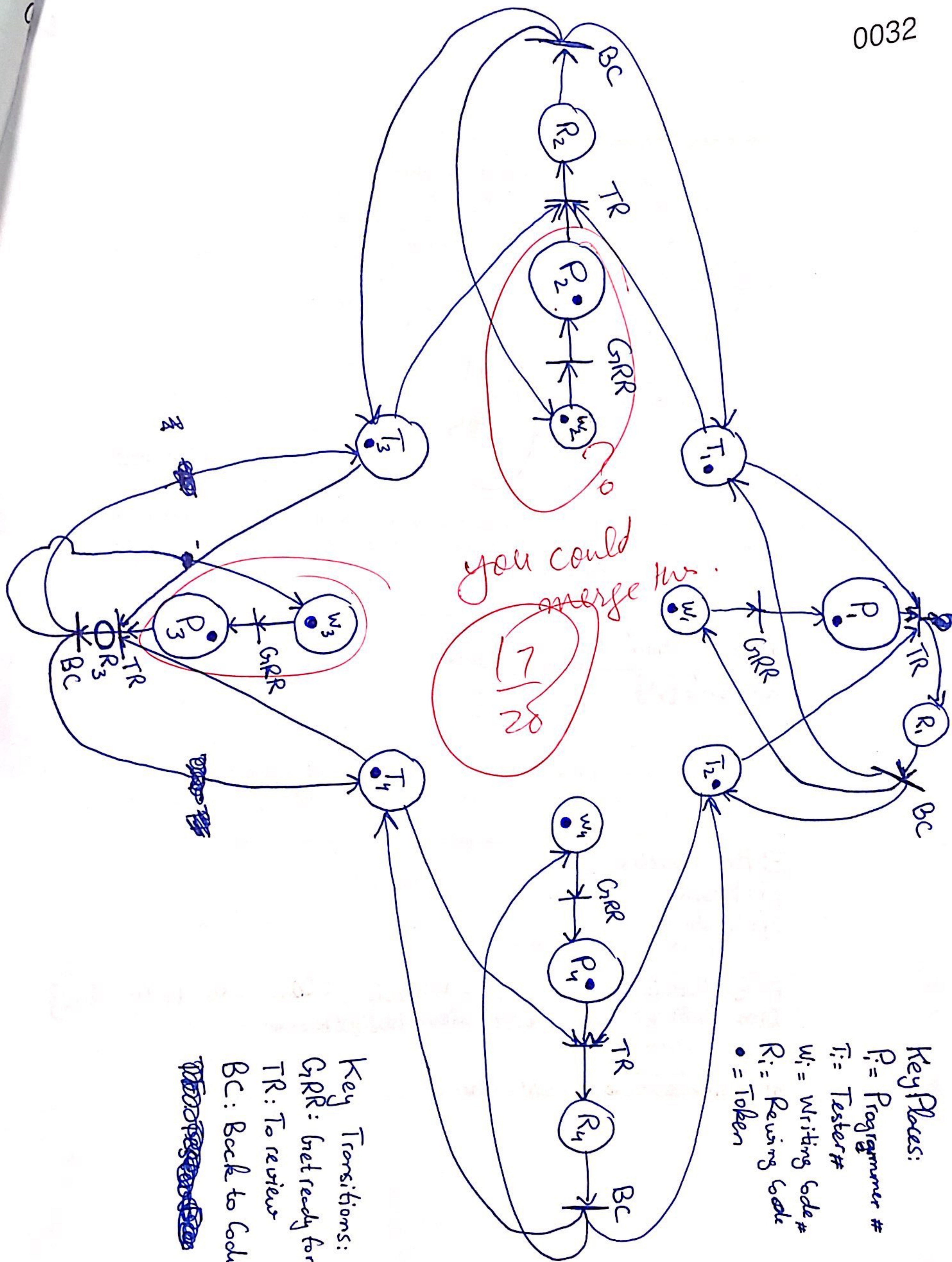
Infosys Inc. is facing problems with reference to huge number of bug reports resulting in project delays due to increased testing and bug fixing time. So the management decides to use a variation of pair programming. Pair programming is a software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in.

Info sys has decided to merge the testing and development teams such that each programmer is paired with two testers. But due to limited number of testers, they have to be shared. Each programmer either programs or reviews his/her code with the testers sitting beside him/her. So when ever both testers are available the programmer stops programming and get its code reviewed. A tester can review code of only one programmer at a time. The programmers and testers are made to sit on a round table to save time.

Draw a neat petrinet for the above problem given that there are 4 programmers and 4 testers. Which means that at any particular time only two programmers can get their code reviewed by the testers. Use a full sheet (landscape view) to draw the petrinet. Label your places and transitions such that they are self-explanatory.



This problem is similar to dining philosophers



Free
Tester

Question # 4 (5+5+5+5+5 Points)

An organization has a system for keeping track of its employees while they are on the premises. Each employee is issued with an active badge which reports their current position to a central database. If the set of all people is Person, and the set of all locations is Location, then the information provided by the system may be described by a relation where is of type Person \rightarrow Location. It is impossible for an employee to be in two places at once, so this relation will be a partial function.

We use a Z schema to describe the structure of Persons' Locations, Whereis contains all information about employees and their location.

PersonLocation _____
Whereis: Person \rightarrow Location _____

21
25

Note: Only write the success scenarios for each operation. Don't use the names of input and output variables that are same as the existing key words. Write necessary pre and post conditions

- a) When a PersonLocation schema is initialized, it contains no information about location of any person, so the value of Whereis should be the empty function. The following schema describes the initial state of a PersonLocation

PersonLocationInit _____
PersonLocation'
Whereis' = $\{\emptyset\}$

5

- b) A successful retrieve operation requires an existing Person as input and provides the corresponding current Location as output. It leaves the system unchanged.

RetrievePersonLocation _____
 \exists PersonLocation
 $p?: \text{Person}$
 $l!: \text{Location}$
 dom
 $p? \in \text{Whereis}$
 $l! = \text{Whereis} \in \{loc: \text{Location} \mid p? \mapsto loc\}$
~~Person' = Person~~
~~Location' = Location~~
~~Whereis' = Whereis~~

3

- c) A successful Update operation replaces the Location stored under an existing Person, and provides no output.

UpdatePersonLocation

Δ Person Location

$p? : \text{Person}$

$l? : \text{Location}$

$\{p? \mapsto l?\} \in \text{Where is}$

~~Where is~~ Where is' = Where is $\oplus \{p? \mapsto l?\}$

Person' = Person ~~Where is~~

Location' = Location

5

- d) A successful delete operation requires that the Person in question exists. A single input is required, and the state of the system will change after the particular person's location will be deleted:

DeletePersonLocation

Δ Person Location

~~Where is~~

~~$l? : \text{Location}$~~

$p? : \text{Person}$

Person in question

$l? \in \text{Where is}$

Where is' = Where is / $\{p? : \text{Person} \mid p \rightarrow l?\}$

Person' = Person ~~Where is~~

Location' = Location / $\{l?\}$

3.5

- e) A successful add operation has a complementary precondition. This time, the Person $p?$ must not be in the domain of PersonLocation. A new entry shall be made containing information about a new Person's Location.

AddPersonLocation

Δ Person Location

$p?: \text{Person}$

$l?: \text{Location}$

dom

$p? \notin \text{Whereis}$

$\text{Whereis}' = \text{Whereis} \cup \{p? \mapsto l?\}$

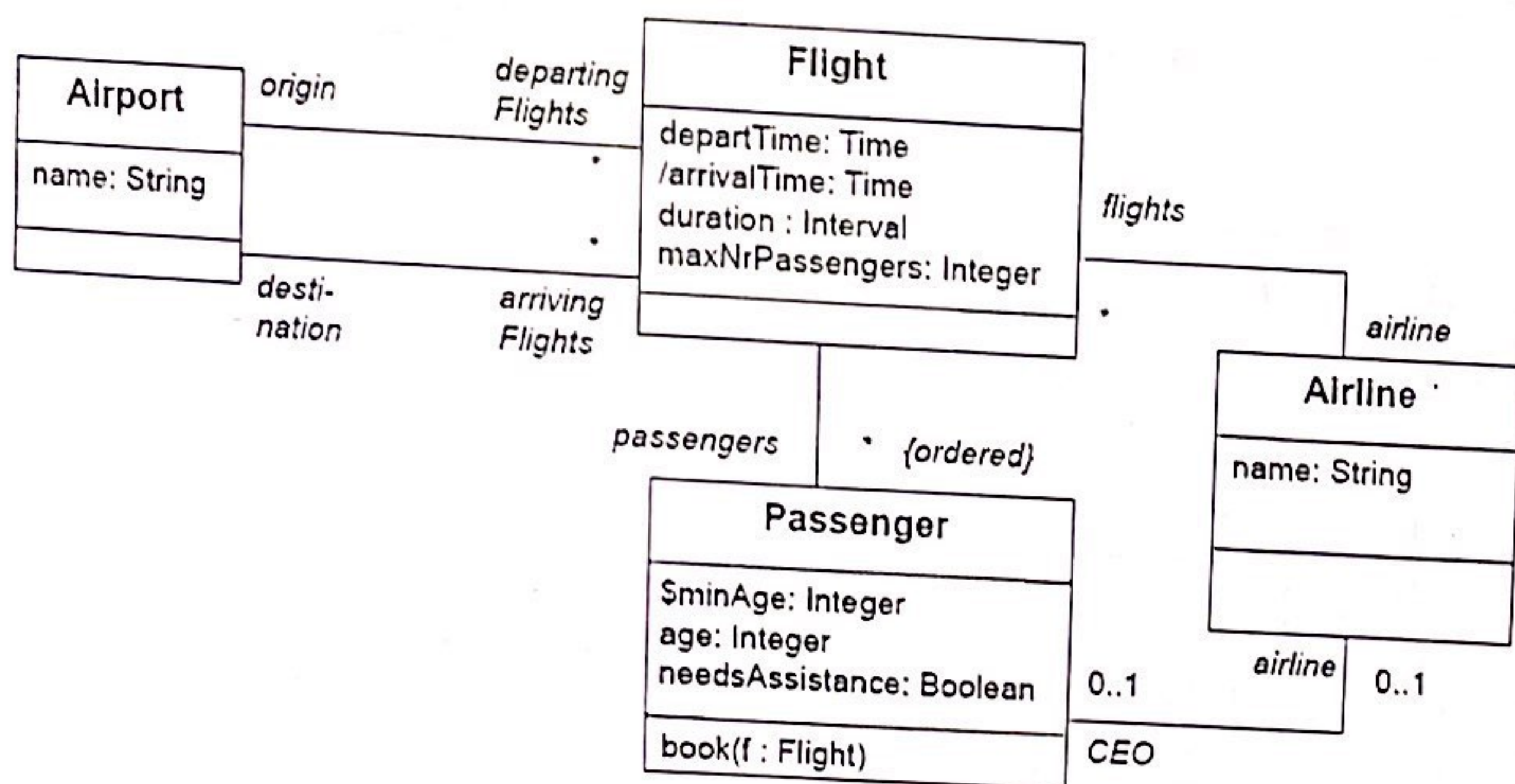
$\text{Person}' = \text{Person} \cup \{p?\}$

$\text{Location}' = \text{Location} \cup \{l?\}$

✓ 4.5

Question # 5 (3+3+3+3+3 Points)

Carefully analyze the following UML diagram. Write OCL constraints keeping in mind the concepts taught in the class.



- a) Age of the CEO of the Airline is not less than 40 years

Context Airline

inv:

self.CEO \rightarrow isEmpty() implies

self.CEO.age ≥ 40

15/15

3

- b) Arrival time of a flight is more than its departure time

Context Flight

inv: $\text{self.departTime} < \text{self.arrivalTime}$

OR
we can
also do

$\rightarrow \text{self.departTime} + \text{self.duration} = \text{self.arrivalTime}$

✓ 3

- c) Origin and Destination of a flight are not same

Context Flight

inv: $\text{self.origin} \rightarrow \text{isNotEmpty()} \text{ implies}$

$\text{self.origin.name} < > \text{self.destination.name}$

3

- d) Count the number of all passengers in a particular flight that need assistance

Context Flight

inv: $\text{self.passengers} \xrightarrow{\text{select}} (\text{self.passengers.needsAssistance} \Rightarrow \text{true}).\text{sum}()$

3

- e) An Airport should have an equal number of departing and arriving flights

Context Airports

inv:

$\text{self.departingFlights.size()} == \text{self.arrivingFlights.size()}$

✓ 3

Good Luck