International Institute of Information Technology, Hyderabad

(Deemed to be University)

SSAD & Project/ICS-261 Monsoon 2018

Question cum Answer Booklet

End Semester Examination - SSAD

Max. Time: 3 Hrs **Max. Marks: 100**

Roll No:_________________ Programme: ________________ Date of Exam:______

Room no: _______ Seat No: ______ Invigilator's Signature:____________________

Special Instructions about the exam:

1. This is a closed-book exam. You may use a one page 8 ½ * 11 inch **hand-written** cheat sheet. Xerox copies or any downloaded notes/slides is not allowed.
2. You are to answer all 7 questions. You have 180 minutes, for a total of 100 points. Credit will be given based on Quality of the content, not Quantity. Voluminous content-free answers not expected

Additional sheet for rough work is allowed: x Yes ☐ No

Marks Table (To be filled by the Examiner)

Q # >								**Name of the Examiner**
Marks								
Q# >								
Marks								
Q# >								
Marks								
Q# >								
Marks								

General Instructions to the students

1. Place your Permanent / Temporary Student ID card on the desk during the examination for verification by the Invigilator.
2. Reading material such as books (unless open book exam) are not allowed inside the examination hall.
3. Borrowing writing material or calculators from other students in the examination hall is prohibited.
4. If any student is found indulging in malpractice or copying in the examination hall, the student will be given 'F' grade for the course and may be debarred from writing other examinations.

Best of Luck

Part 1- Concepts (45 points)

1. In the context of initial discussions with a client for a software project: (12 points/ 3 points each part)
    a. List any three examples of the kinds of non-functional requirements: Reliability issues, Performance issues, & Security. (Other examples will also be considered and awarded marks if they are correct.)

    b. As a requirement specification, what is wrong with this requirement: "Library member names should be stored in a sorted descending order"

       This requirement specification is wrong as it overspecificies the requirements. It is addressing "how to" aspects of the solution and therefore, it is restricting the solution space for the designer.

    c. If you are building an food delivery app, state any one *user story* you may define for this app. (Hint: Per SRS best practices)

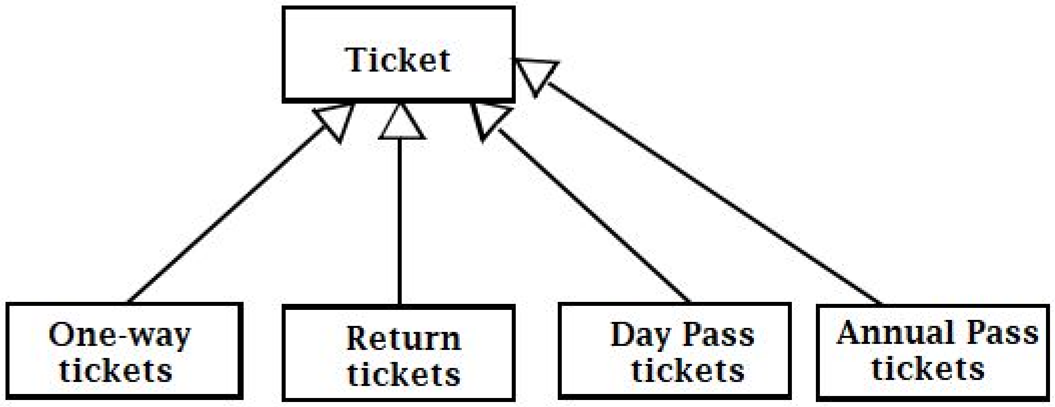
       As a user, I can choose the items I want to order so that the chosen items can be delivered to my doorstep.

       (Other user stories will also be considered and awarded marks if they are correct. A user story has to be from the perspective of the user.)

    d. This requirement is ambiguous. Why? "*The user shall be able to toggle between displaying and hiding all search results in the page with the activation of a triggering condition.*"

       The above requirement is ambiguous as the "triggering condition" is not specified in any way and has been left for design.

2. Design/Model considerations when designing a software system: (12 points/ 3 points each part)
    a. For the bus ticket booking app, draw a possible inheritance model for types of tickets using UML. Use at least 4 types of tickets



(Other types of tickets(like sleeper etc.) will also be awarded marks if they are correct.)

b. Very briefly explain *Data Abstraction* & *Information Hiding* using one example.
1 mark for each of the definition/explanation and 0.5 marks each for the example. *(2x1 + 2x0.5 = 3)*

Data Abstraction:

- A clear separation between the abstract properties of a data type and the concrete details of its implementation.
- The abstract properties are those that are visible to client code that makes use of the data type, while the concrete implementation is kept entirely private.
- For example, consider an abstract data type called lookup table which supports looking up values corresponding to a given key. Such a lookup table may be implemented: as a hash table, a binary search tree, etc. As far as client code is concerned, the abstract properties of the type are the same in each case.

Information Hiding:

- Information hiding is the process of hiding the details of an object or function.
- It effectively decouples the calling code from the internal workings of the object or function being called, which makes it possible to change the hidden portions without having to also change the calling code.
- For example, a car manufacturer may have a luxury version of the car as well as a standard version. The luxury version comes with a more powerful engine than the standard version. The engineers designing the two different car engines, one for the luxury version and one for the standard version, provide the same interface for both engines. Both engines fit into the engine bay of the car which is the same between both versions, but their internal workings are different.

  (Any other example/explanation would also be awarded marks if they are correct.)

c. In Refactoring, what is 'Extract Method'?

"Extract method" tells us to extract lines of code from a long method and make it a separate method.

d. Singleton pattern & Abstract factory pattern/Factory method pattern are two examples of *Creational Patterns (design patterns)*.

Marks are given for writing any two examples of creational patterns.

3. Solution Testing (12 points/ 3 for each part)
    a. As exhaustive testing is practically impossible, *Selective Testing* is a general practice. function,statement and path coverage is an indicator of effective testing when using *Selective Testing*.

    b. List the Equivalence Classes for testing a field capturing weight (in KGs) of a person.
    [-infinty, 0], [0, 90], [90,infinty]

    c. Differentiate *Acceptance testing* and *System testing*.

- System testing is end to end testing performed to check if the software meets the specified requirements.
- System testing is performed by developers and testers.
- System Testing can be both functional and non functional testing.

- Acceptance testing is functionality testing performed to check if the software meets the customer requirements.
- Acceptance testing is performed by independent set of testers and also the stakeholders, clients.
- Acceptance testing is pure functional testing.

    d. Testing or Inspection? Which comes first (1 point). Why (2 points).

    INSPECTION -- 1 mark

The main reason for this is that inspecting allows us to quickly get rid of many defects even before developer testing. - 2 marks

4. Development Processes (12 points/ 3 for each part)
    a. Define *Work breakdown structure.*

- Work Break Down Structure – a check list of the work that must be accomplished to meet the project objectives.
- The WBS lists the major project activities and those departments or individuals primarily responsible for their completion.
- Types

b. Spiral model is a meta model. Explain. (Any 3 points)

Subsumes all discussed models:

- A single loop spiral represents waterfall model. Retains the step-wise approach of the waterfall model.
- Uses an evolutionary approach -- iterations through the spiral are evolutionary levels.
- Enables understanding and reacting to risks during each iteration along the spiral.
- Uses prototyping as a risk reduction mechanism

c. The three types of backlogs in typical Scrum development process are:

- Product Backlog
- Release Backlog
- Sprint Backlog

d. Exact estimate is an oxymoron. Why.

Estimate can never be exact, it is just a prediction which is dependant on a lot of uncertain factors and hence cannot be predicted exactly.

Part 2: Long questions

5. Case study 1- Bus ticket System's Requirements (15 points)

Your company has been engaged to develop an automated bus ticket reservations system (ABTS) for FlyWay Bus Services. Passengers can make reservations at automated reservation machines (ARMs) that are located throughout cities served by FlyWay. A reservation will include the bus number, source city, destination city, departure date and time, reservation type (first class, sleeper, sitting), a seat number, and the price of the ticket. Once reservations are made and credit card information is verified, tickets are printed for the passenger. Passengers can also cancel or change reservations after they are made. All flight data is entered by a designated data entry clerk. (**Note:** You may make simple assumptions about this problem, based on your own knowledge about bus reservation systems.)

a) Write the detailed textual use case (step-by-step) for a passenger successfully using the ABTS to book a reservation. (5 points)

Use Case Name:	Booking a reservation
Overview(Optional):	Using the ABTS platform to make a bus reservation

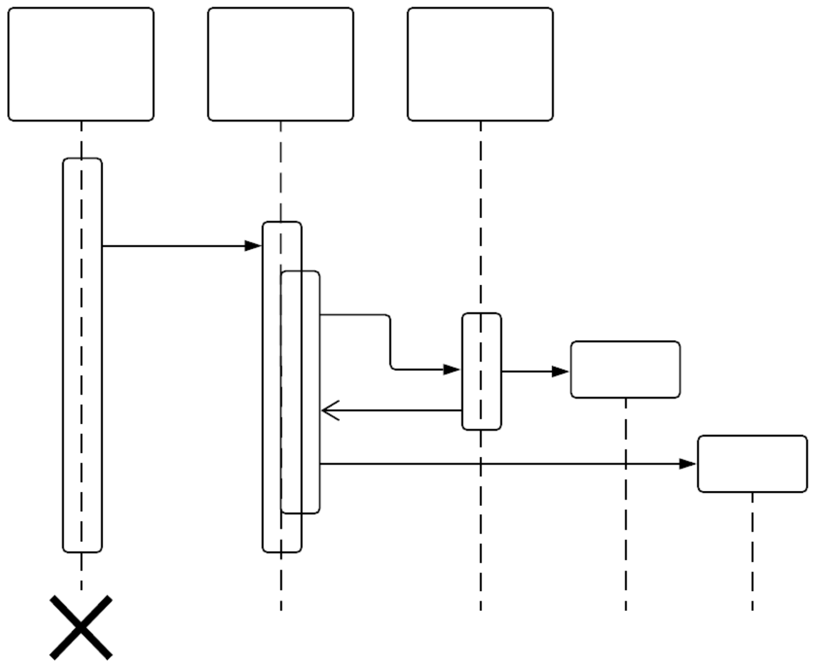
Type(Optional, as given in Question):	Primary
Actors:	Passenger, ARM, Data entry clerk
Pre-Condition:	- Src, dest served by Flyway - Appropriate date and time entered
Main Flow:	1) Enter details into the ARM 2) Verify availability (comparison to details in the database by the data entry clerk) 3) Proceed to booking (enter credit details/payment options) 4) Once payment has been verified, update database to reflect the booked ticket. 5) Display ticket confirmation on the ARM, and provide option to print or email ticket
Alternate Flow:	3) If payment is not successful, go back to (3), and choose other payment option. 4) In case the ticket is unavailable for the given combination, go back to (1), and prompt the user for another set of parameters
Post Condition:	Ticket has been booked and relevant data has been updated

b) Draw a UML Design class diagram for the **ABTS** showing the classes, relationships, and multiplicities between classes (6 points)

- **Must** have the following classes: (4 points)
    - Ticket
    - Clerk
    - Passenger
    - ARM (*or equivalent*)
- Show multiplicities for **all** classes in the diagram (1 point)
- Interactions should be labeled above the connections (1 point)
- The diagram should have a coherent structure, additional classes are allowed so long as they represent the system in a real world scenario. (-2 points for illogical interaction schema)

c) Using your class model, create a UML sequence diagram for a passenger cancelling a booking previously made. (4 points)

- Passenger, ARM, Database actors **must** be mentioned (2 points)
- Lifelines for each actor should be clearly marked (1 point)
- Appropriate interaction labelling above the arrows and indication of the flow (1 point)
- **Any other diagram than the type shown below should be given a 0.**



6. Case study 2- Usecases (15 points)

36Moto is a Custom Motor Cycles company, that is engaging your company to build an operations software system for them. 36Moto undertakes custom motor cycle projects, where each project is unique. Client has custom specifications, there is a design phase, build phase, integrate and delivery. The operations include showcase of previous works & capabilities, typical build process, enquiry, project estimates, commencement, design, procurement, build, delivery and post delivery support. Each bike project goes thru these phases of activity. The information for each phase must be maintained. There are customers. Each customer may have multiple projects (bike builds). Each build has the bike specifications. Each build has the Bill of Materials (inputs to be procured). There are suppliers (vendors) that provide varous input materiials. There are purchase orders for the procurement. Each build project has a project plan.

a) Draw the usecase diagram for the primary usecase of tracking a custom bike build from order thru delivery?

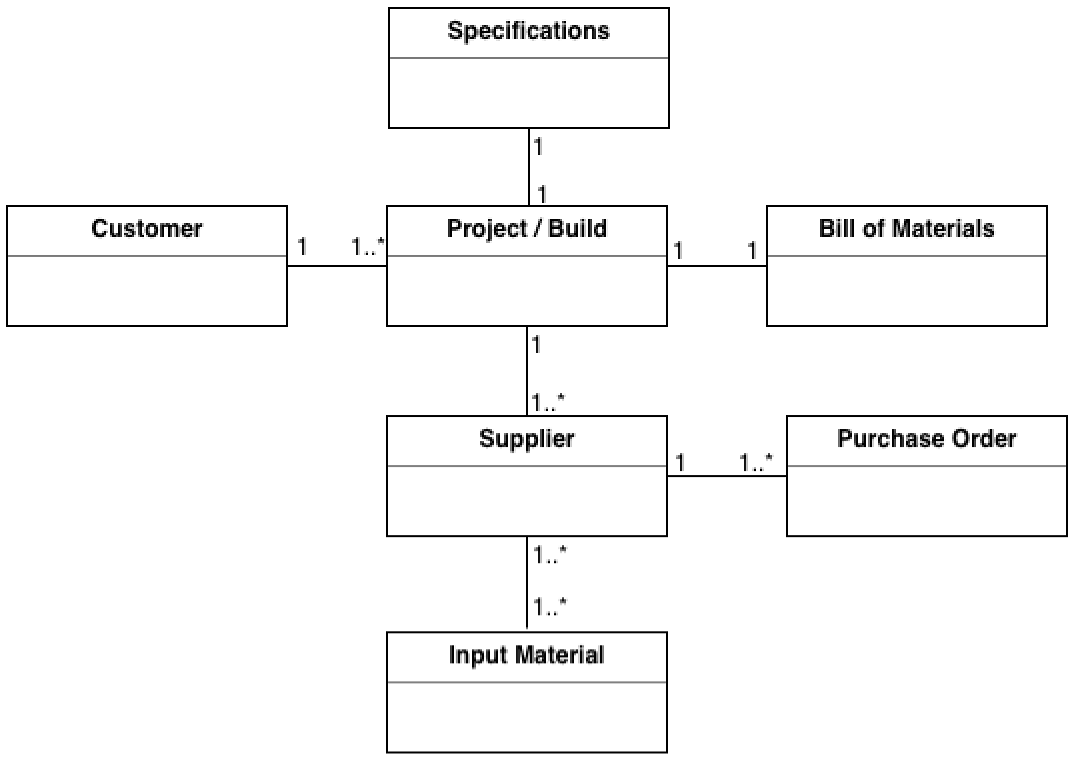
Question changed to write Use Case in textual form

Structure: 2M
Stated Context: 1M
Main Flow: 3M
Actors: 2M
In Tabular form (Marks given for original question if done correctly with appropriate assumptions):

Use Case Name:	Tracking Order
Overview(Optional):	Tracking a Custom Bike Build from order til delivery

Type(Optional, as given in Question):	Primary
Actors:	Client, Suppliers
Pre-Condition:	Customer Placing Order
Main Flow:	6) Take Down Customer Specifications 7) Project Plan is Created 8) Design the Bike 9) Request Supplier for Input Materials 10) Exchange of Purchase Order and Bill of Materials 11) Bike is built, and parts are integrated 12) Delivery
Alternate Flow:	3) If Design not feasible, go back to (1) and re-evaluate. 4) If Supplier doesn't have required materials, revisit design for alternatives or contact customer to change specifications.
Post Condition:	Customer has received Bike

b) Draw the UML domain model for this solution.



Marks Allotting:
3M for 6/7 classes to represent information
2M for correct multiplicity
2M for overall correctness / structure

7. Case study 3- State diagrams (25 points)

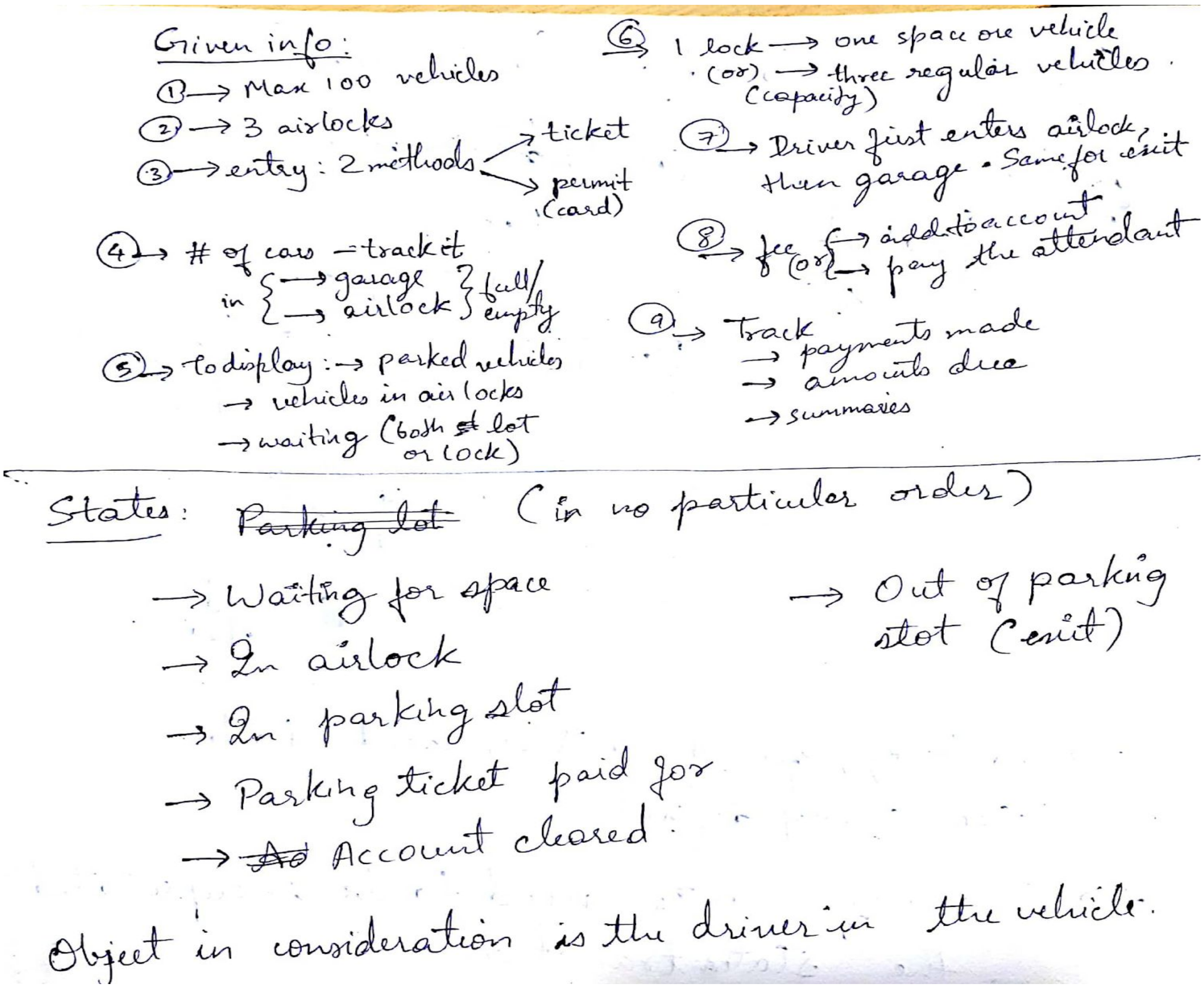
SoftModel Inc. has been contracted to build a system for parking garages attached to the Space Stations of the 22nd century. The first parking garage to be built will be attached to "Space Station Siren" which orbits Titan, Saturn's moon. The parking garage will be able to park up to 100 space vehicles of all types. It will be have 3 air locks which allow the space vehicles to move from airless space into the pressurized atmosphere of the space station. Each air lock can fit either one space ore vehicle or 3 regular (passenger or military) space vehicle. Future parking garages may have more (or fewer) parking spaces and a different number and size air lock.

The parking technology is similar to those currently used on Earth. Drivers can enter the garage with a permit that can be swiped with a card reader. They can also get a ticket by pressing a button at the entrance. The vehicles have automatic arms to get the ticket inside. You don't want to open a window and let the air out. The ticket has a date and time-stamp indicating the date/time of entering the garage. After swiping the permit or getting a ticket, the driver enters an air lock (if there is room) and then enters the garage itself.

The system to be developed by SoftModel should keep track of the number of cars currently in the garage, and in each air lock. It displays signs indicating whether or not the garage is full, and whether or not each air lock is full. Drivers with tickets pay the attendant at a gate before leaving the garage; drivers with permits have a parking fee added to their account when they swipe their permits upon exiting from the garage. Drivers must pass through an air lock to exit the garage. An operator console displays the status of the system including the number and types of (1) parked vehicles, (2) vehicles in air locks, and (3) vehicles waiting for space in the lot or space in the air locks.

The system must keep track of the payments being made, and the amounts due from permit holders. It must also provide summaries to authorized personnel regarding peak garage hours and the number of permits used.

Design a single **state diagram** for the above system. Please state your assumptions clearly (if any).



Given info:

① → Max 100 vehicles

② → 3 airlocks

③ → entry : 2 methods → ticket / → permit (card)

④ → # of cars - track it in { → garage, → airlock } full/empty

⑤ → To display : → parked vehicles
→ vehicles in air locks
→ waiting (both ~~at~~ lot or lock)

⑥ → 1 lock → one space one vehicle (or) → three regular vehicles. (capacity)

⑦ → Driver first enters airlock, then garage. Same for exit

⑧ → fee (or) { → add to account, → pay the attendant }

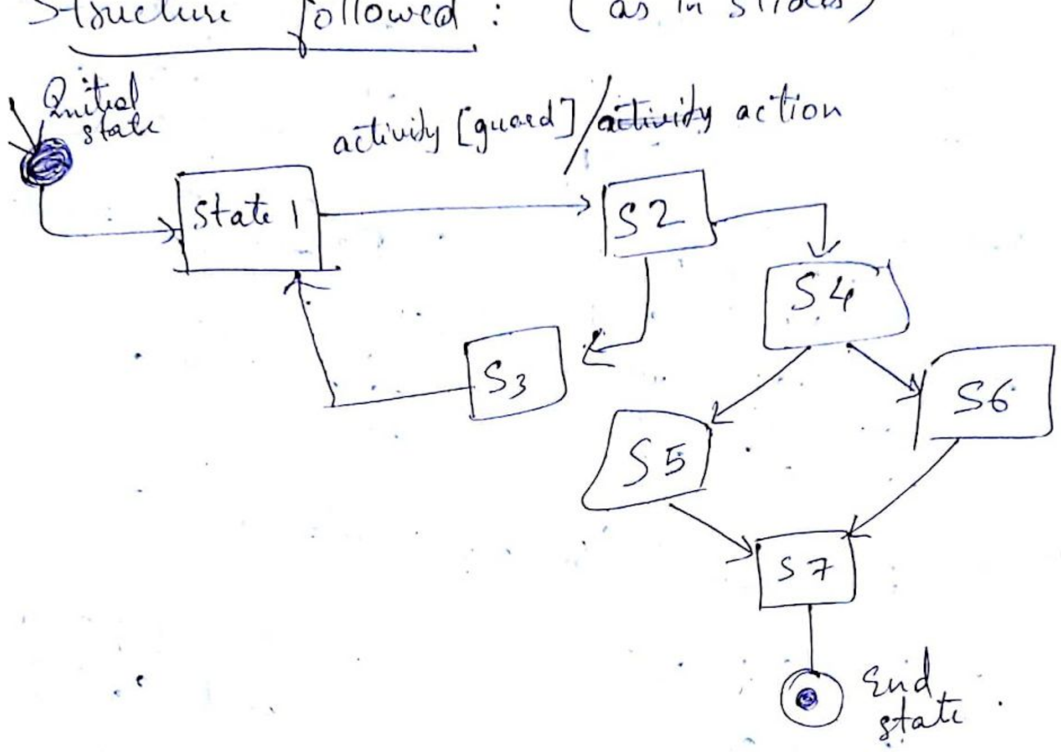
⑨ → Track
→ payments made
→ amounts due
→ summaries

States : ~~Parking lot~~ (in no particular order)

→ Waiting for space
→ In airlock
→ In parking slot
→ Parking ticket paid for
→ ~~Ac~~ Account cleared
→ Out of parking slot (exit)

Object in consideration is the driver in the vehicle.

Structure followed : (as in slides)



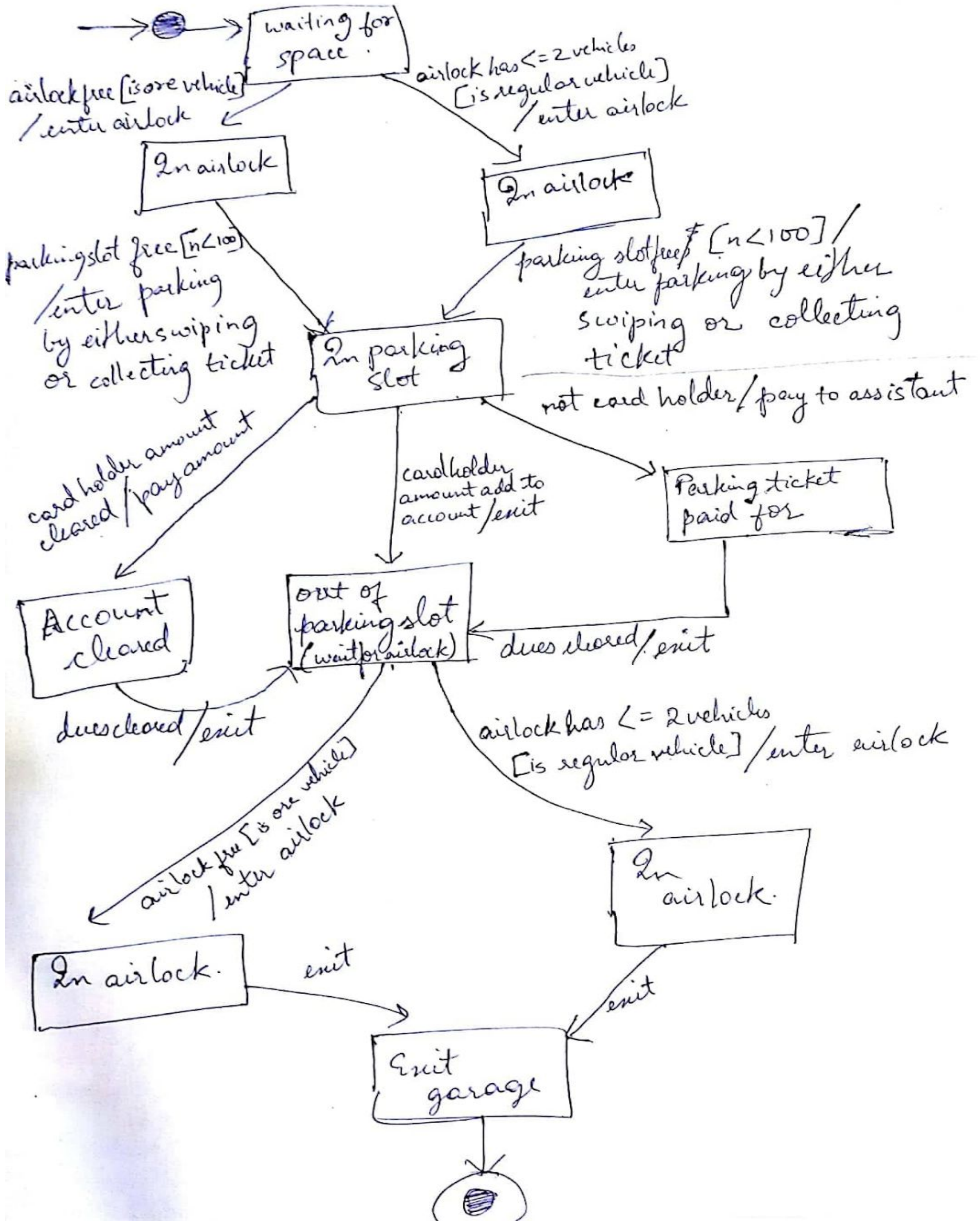
⇒ Activity, guard, action need to be ~~clo~~ clearly defined.

⇒ Flow of events need to be encapsulated in the ~~states~~ state transition.

⇒ Marks will be awarded if all the information is represent; ie. the state diagram is self-explanatory.

⇒ Understanding of the concept of object, state and transitions will be checked for.

The state diagram shown below is an example. Marks will be awarded for correct states and objects, where assumptions are stated clearly.
Explanation is not expected. The state diagram alone should be able to capture all the concepts.
Marks: 10M for structure, 5 for correct object choice, 10 for all the states mentioned in the question.


waiting for space
airlock free [is one vehicle] / enter airlock
airlock has <= 2 vehicles [is regular vehicle] / enter airlock
In airlock
In airlock
parkingslot free [n<100] / enter parking by either swiping or collecting ticket
parking slot free [n<100] / enter parking by either swiping or collecting ticket
In parking slot
not card holder / pay to assistant
card holder amount cleared / pay amount
cardholder amount add to account / exit
Parking ticket paid for
Account cleared
out of parking slot (wait for airlock)
dues cleared / exit
dues cleared / exit
airlock has <= 2 vehicles [is regular vehicle] / enter airlock
airlock free [is one vehicle] / enter airlock
In airlock.
In airlock.
exit
exit
Exit garage