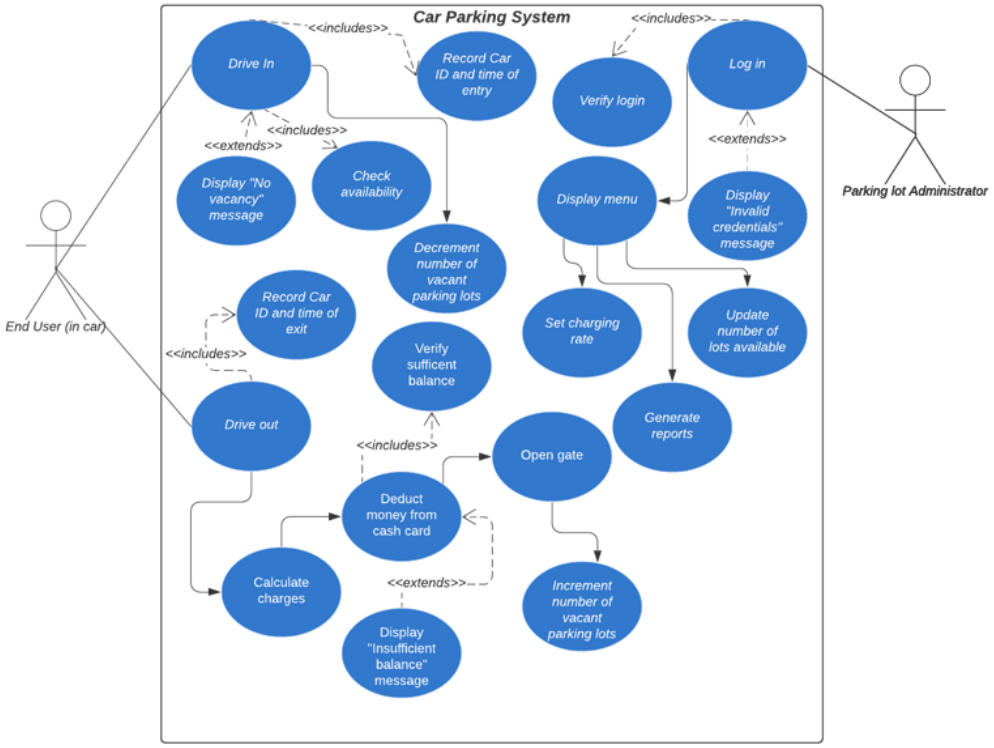
1 (a)

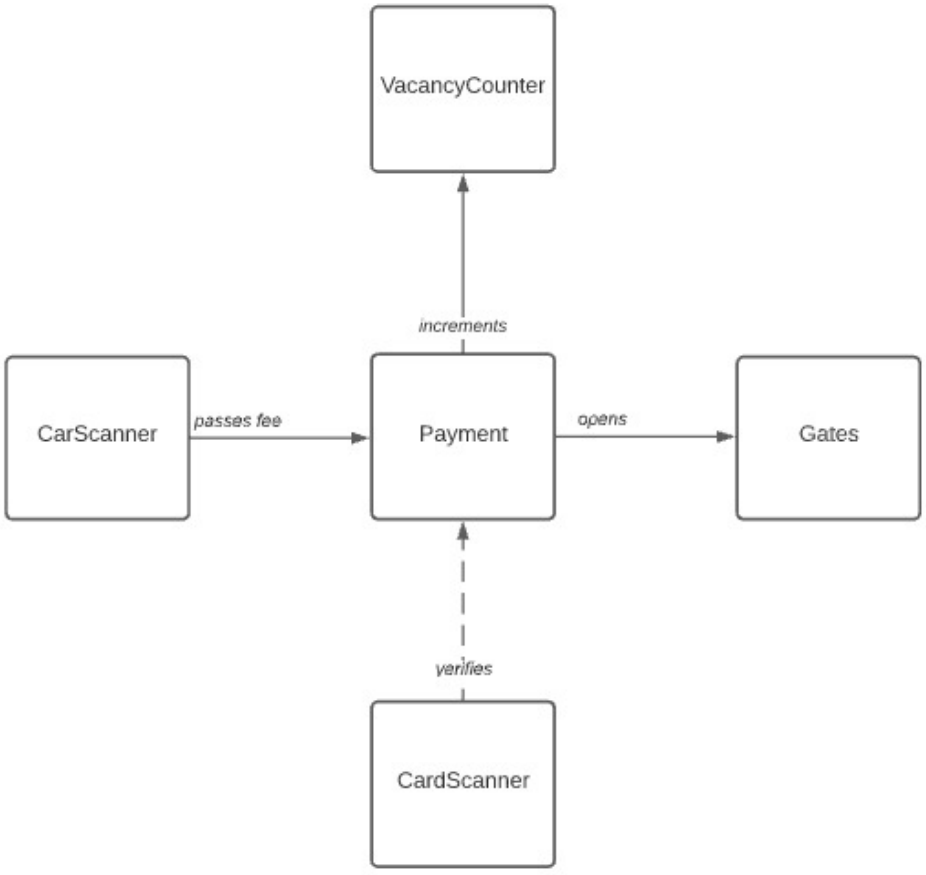


(b)

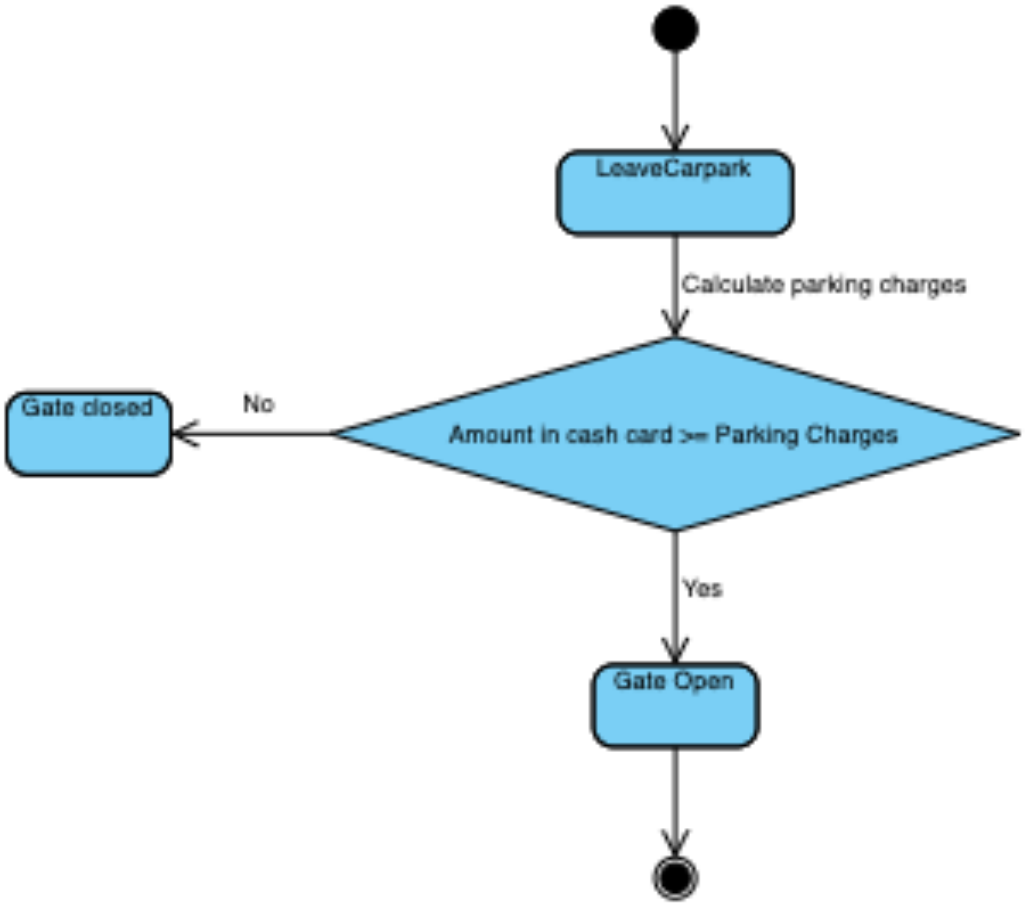
Use Case Name	Leaving the parking lot
Participating actor(s)	Driver/Car
Flow of Events	1. Car drives towards exit 2. System records car ID and time 3. System calculates charges based on rate and time of entry and exit 4. User inserts cash card 5. System verifies sufficient balance 6. System deducts money 7. System opens gate 8. System increments number of vacant lots

Alternative flow of events	1. Car drives towards exit 2. System records car ID and time 3. System calculates charges based on rate and time of entry and exit 4. User inserts cash card 5. System checks balance and finds it insufficient 6. Gate does not open 7. User has to refill cash card and pay to leave
Entry conditions	User tries to leave the parking lot
Exit conditions	User pays the parking fee

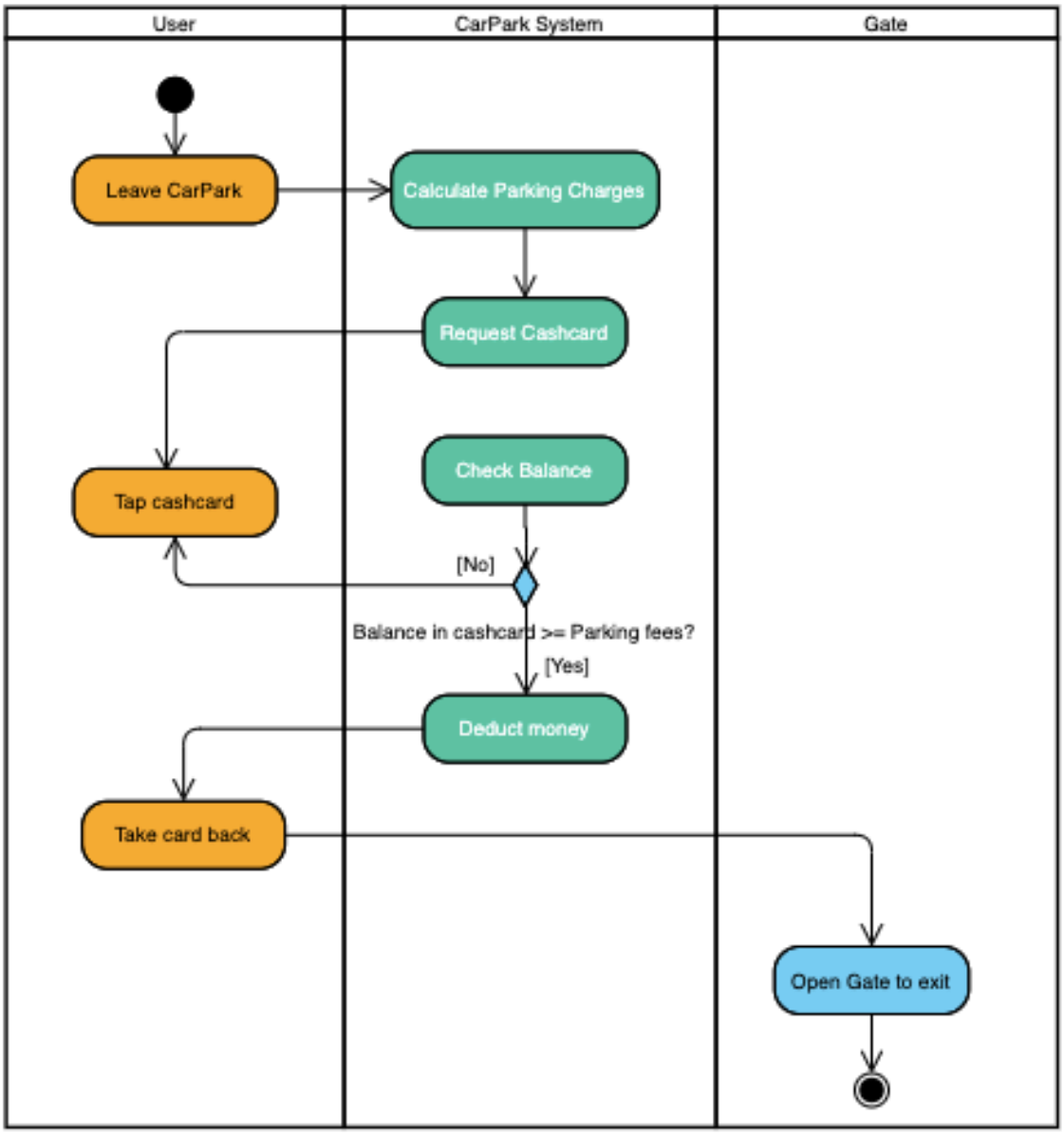
(c)



2 (a)



(b)



(c) Task A: 4 weeks – No prereq
Task B: 2 weeks – After A finishes
Task C: 3 weeks – After A finishes
Task D: 3 weeks – After B and C finishes
Task E: 5 weeks – After B and C finishes
Task F: 2 weeks – After D and E finishes
Task G: 1 week – After D and E finishes
Task H: 2 weeks – After F and G finishes

Forward Path Analysis

Task – Earliest Start – Earliest Finish
A – Week 1 beg – Week 4 end
B – Week 5 beg – Week 6 end
C – Week 5 beg – Week 7 end
D – Week 8 beg – Week 10 end
E – Week 8 beg – Week 12 end
F – Week 13 beg – Week 14 end
G – Week 13 beg – Week 13 end
H – Week 15 beg – Week 16 end

Backward Path Analysis

Task – Latest Start – Latest Finish
A – Week 2 beg – Week 5 end
B – Week 6 beg – Week 7 end
C – Week 5 beg – Week 7 end
D – Week 10 beg – Week 12 end
E – Week 8 beg – Week 12 end
F – Week 13 beg – Week 14 end
G – Week 14 beg – Week 14 end
H – Week 15 beg – Week 16 end

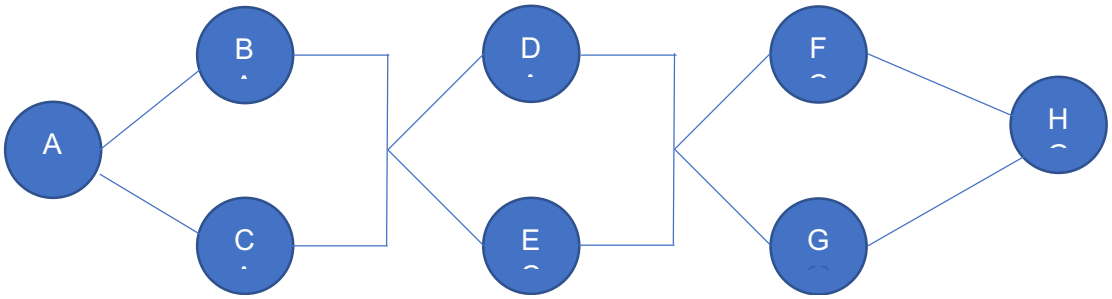
Slack time = Latest Start – Earliest Start

Slack Times:
A – 1 week
B – 1 week
C – 0 weeks
D – 2 weeks
E – 0 weeks
F – 0 weeks
G – 1 week
H – 0 weeks

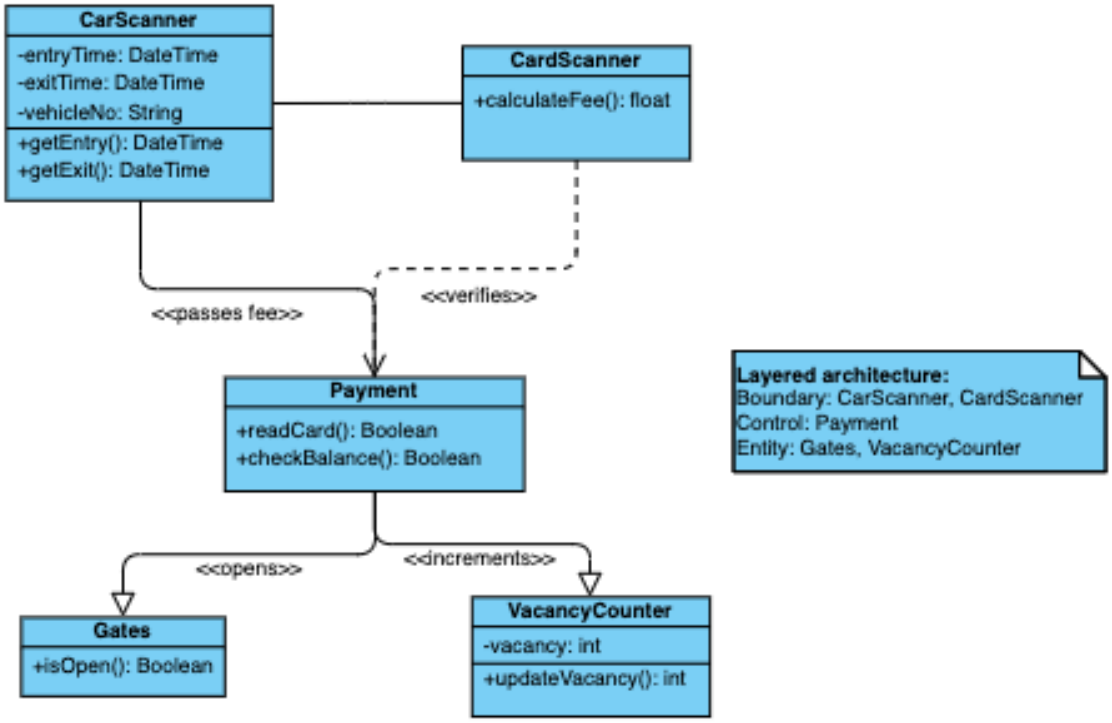
Critical path : C -> E -> F -> H

Slack Time of D = 2 weeks

Activity diagram:



3 **(a)** (i) Layered architecture with three layers: Entity, Boundary and Control



(ii) MVC Design Pattern:

- ➔ Simultaneous Development
- ➔ Multiple views for a model – Model can have multiple views
- ➔ High cohesion: MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together
- ➔ Low coupling: The very nature of the MVC framework is such that there is low coupling among models, views or controllers.

(b) (i) (1) In JDK1.1 and later versions, the event processing model adopts the delegation event model (Delegation Event Model, DEM) based on the observer pattern.
In DEM, the publisher of the event is called the event source (Event Source), and the subscriber is called the event listener (Event Listener). In this process, the event object (Event Object) to transfer event-related information, event processing can be implemented in the implementation class of the event listener, so the event listener object can also be called the event processing object.
Event source object, event listener object (event processing object) and event object constitute the three elements of the Java event processing model.
(2) In addition to the event processing in AWT, the Java language parsing XML technology SAX2 and the Servlet technology event processing mechanism are based on DEM, and they are all applications of the observer mode.
(3) The observer mode is widely used in software development. For example, an e-commerce website can send users multiple discounts on products after performing the sending operation. In a team battle game, The sacrifice of teammates will give all members tips and so on. Observer mode can be used in any one-to-one or one-to-many object interaction scenes.

(ii) Carpark vacancy availability tracking using Façade pattern
JDBC, RSS, DB are three API elements of Façade pattern
(iii) Less accuracy in load management; Influence of cross sectional variations; Tension during compressive testing are some disadvantages.

[Note: Not sure of answer, read from research paper, attached snippet]

For bored concrete piles, auger piles and caissons the dynamic load testing method has some disadvantages and is less suitable and statnamic load testing is the preferred method. The most important reasons for the preference of statnamic load testing in the case of cast in situ piles are:

1. **Accuracy in load measurement**
STN is not dependent on pile material and cross section properties
2. **No influence of cross sectional variations**
STN results are not influenced by cross sectional variations over the pile length
3. **No tension during compressive testing**
STN long duration loading will keep pile under constant compressive pressure
4. **Concentric loading**
Easy placement of STN loading device in center of the pile
5. **Pile and soil response closer to static**
With STN the pile moves as one unit, similar to static load tests. Stress wave phenomena can be neglected resulting in a simple method of analysis

For driven piles both DLT and STN methods can be applied reliably and each has its advantages and disadvantages. A big economic advantage for DLT can be the use of the production rig for testing. A big advantage for STN is the fact that maximum available energy can be used to mobilize capacity and that that testing does not have to be stopped when tensional stresses become too high like with DLT.

4 **(a)** (i)

Equivalence classes for:

1) Vehicle number: 1 valid class for ASCII values and 1 invalid class
2) Vehicle type:
   Valid classes: [Car], [Bus], [Motorcycle], [Lorry]
   Invalid classes: [Unknown]
3) Entry time:
   Valid class: [Datetime.year <= 2020 && Datetime.year >= 2000]
   Invalid class: [Datetime.year >= 2020 || Datetime.year <= 2000]
4) ID number:
   Valid class: [ID >= 10^{10} && ID < 10^{11}]
   Invalid class: [ID <= 10^{10} || ID <= 10^{11}]

ii)

1) Vehicle number:
   Lower boundary value: 0
   Value just below lower boundary: -1
   Value just above lower boundary: 1
   Upper boundary value: 255
   Value just below upper boundary: 254
   Value just above upper boundary: 256
2) Vehicle type: NA

3) Entry time:
   Lower boundary value: 2000
   Value just below lower boundary: 1999
   Value just above lower boundary: 2001
   Upper boundary value: 2020
   Value just below upper boundary: 2019
   Value just above upper boundary: 2021
4) ID number:
   Lower boundary value: 10^{10}
   Value just below lower boundary: 10^{10}-1
   Value just above lower boundary: 10^{10}+1
   Upper boundary value: 10^{11}
   Value just below upper boundary: 10^{11}-1
   Value just above upper boundary: 10^{11}+1

iii)

1) Vehicle number:

Equivalence class	Test case	Validity
ASCII values	AB345	VALID
Non ASCII values	£	INVALID

2) Vehicle type:

Equivalence class	Test case	Validity
Car	Car	VALID
Bus	Bus	VALID
Motorcycle	Motorcycle	VALID
Lorry	Lorry	VALID
Unknown	Unknown	INVALID

3) Entry time:

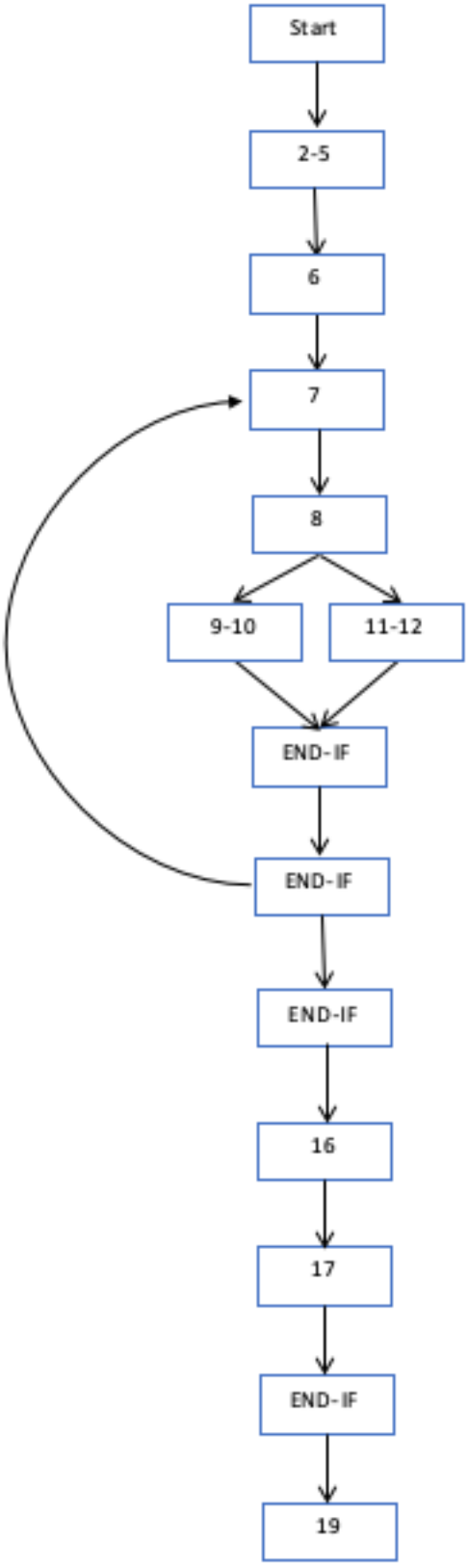
Boundary Value	Validity
Lower boundary value: 2000	VALID
Value just below lower boundary: 1999	INVALID
Value just above lower boundary: 2001	VALID
Upper boundary value: 2020	VALID

Value just below upper boundary: 2019	VALID
Value just above upper boundary: 2021	INVALID

4) ID number:

Boundary Value	Validity
Lower boundary value: 10^{10}	VALID
Value just below lower boundary: 10^{10}-1	INVALID
Value just above lower boundary: 10^{10}+1	VALID
Upper boundary value: 10^{11}	VALID
Value just below upper boundary: 10^{11}-1	VALID
Value just above upper boundary: 10^{11}+1	INVALID

(b) (i)


Start
2-5
6
7
8
9-10
11-12
END-IF
END-IF
END-IF
16
17
END-IF
19

(ii) Cyclomatic complexity = Edges – Nodes + 2 = 14 – 14 + 2 = 2

Solver: Kamakshi Asuri Simhakutty (kamakshi001@e.ntu.edu.sg)