COMSATS University Islamabad, Lahore Campus

Terminal Exam – FALL 2024

Course Title:	Formal Methods				Course Code:	CSE356	Credit Hours: 3(3,
Course Instructor/s:	Dr. Farooq Ahmad, Dr. Junaid Akram, Iqra Obaid				Programme	BS Software Engineering	
Semester:	4th	Batch:	FA22-BSE	Section:	A, B, C, D	Date:	June 12, 2024
Time Allowed:	180 Minutes				Maximum Marks:		50
Student's Name					Reg. No.		

Important Instructions / Guidelines:

- **Answer all questions on the answer paper provided to you.**
- **Do not give multiple answers to a question. Clearly cross out what you do not want me to read.**
- **Do not use lead pencil.**
- **Do show all intermediate steps while solving a question.**
- **Turn off your mobile phone, no silent mode.**
- **Return the question paper alongwith answer sheet.**

Question 01: **[Marks: 6+4=10]**

CLO: < 2 > ***Bloom Taxonomy Level: <Applying>***

Consider the following banking system as a case study, to write formal specifications.

"Consider a system that record's people's birthdays, and can issue reminder when the day comes round. Here known is the set of names with recorded birthdays while birthday returns the birthday associated with a particular name"

BirthdayBook

$known : \mathbb{P}\ NAME$

$birthday : NAME \pfun DATE$

$known = \text{dom}\ birthday$

Please formally specify the following informal description of the operations of a banking system by using Z specification schema's writing method:

a) **Add a Birthday:** Each person can have only one birthday. If the input name is already known to the system state is not changed and the result already_known is returned otherwise add the new birthday is added to the system and the result ok is returned.

b) **Find Birthday:** This operation finds the birthday of the person known to the system

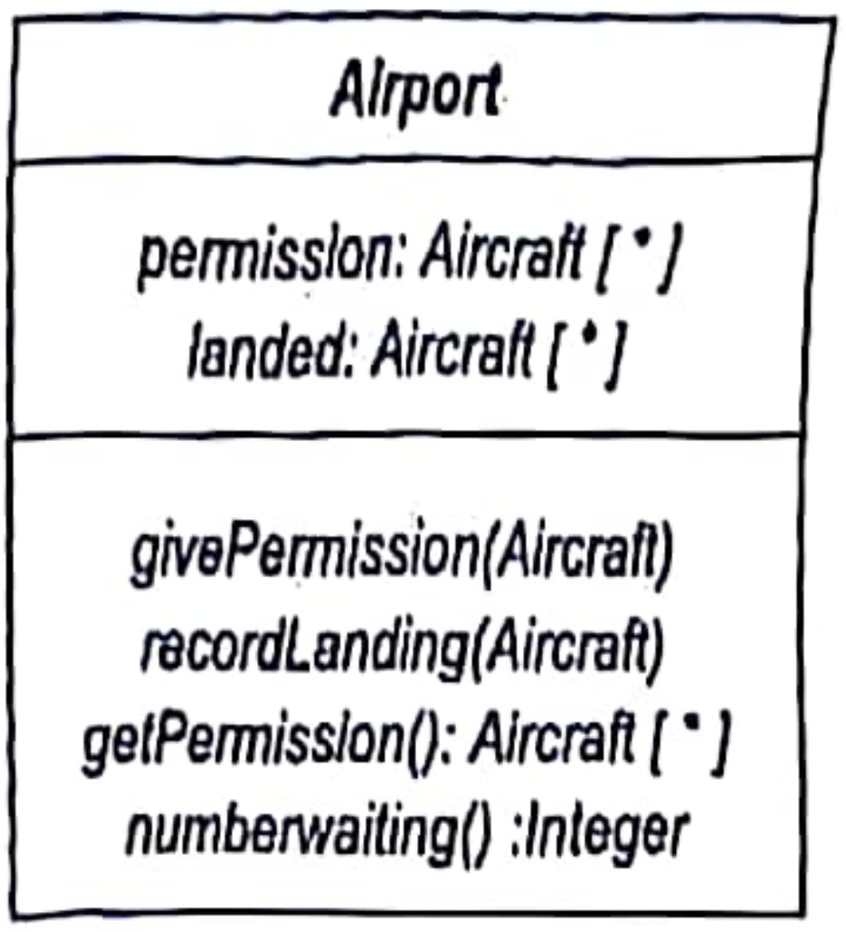
Question 2: **[Marks: = 10]**

CLO: < 3 > ***Bloom Taxonomy Level: <Applying>***

Formally specify the *Airport* class in Z language.

A system that keeps track of aircraft that are allowed to land at a particular airport. Aircraft must apply for permission to land at the airport prior to landing. When an aircraft arrives to land at the airport it should only have done so if it had previously been given permission. The invariant property for the system is the landed aircraft are those who have the permission. Assume that the airport can land 20 aircarfts that could be landed at any time?

The UML specification of the *Airport* class is given below.



i. **givePermission**: records the fact that an aircraft has been granted permission to land.
ii. **recordLanding**: records an aircraft as having landed at the airport.
iii. **getPermission**: returns the aircrafts currently recorded as having permission to land.
iv. **numberWaiting**: returns the number of aircrafts granted permission to land but not yet landed

a) Define the State Schema of the given system. (2 marks)
b) Write the Operational Schemas for the given four operations. (2*4 = 8 marks)

Question 03: *[Marks: 1 + 1 + 2 * 4 = 10]*
CLO: < 4 > *Bloom Taxonomy Level: <Applying>*
Formally specify the system in **VDM-SL.**

Consider a system that records the current mode of an industrial robot, which can either be working, idle or broken.

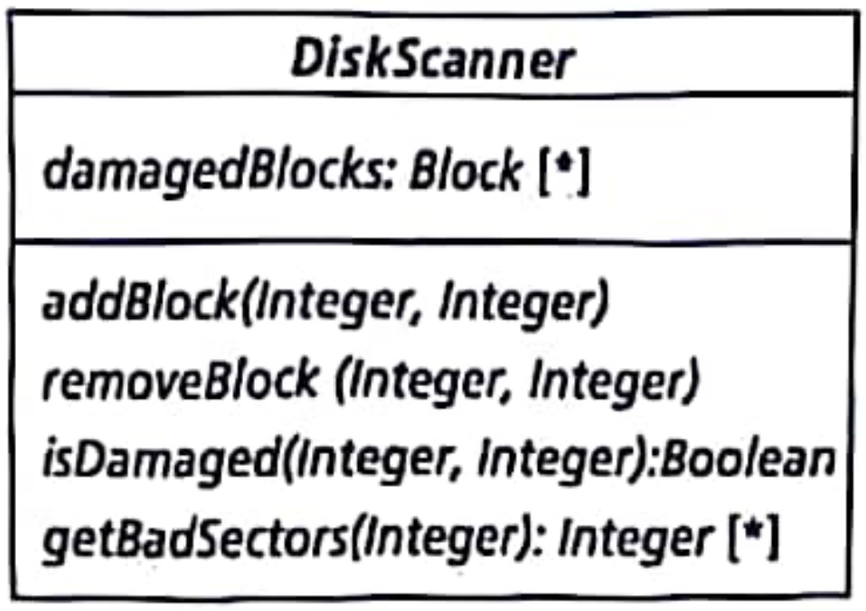
(a) Declare a type, Mode, for use in the specification.
(b) Write the specification of the state of the system, including an initialization function that ensures that the robot is set to idle when the system first comes into existence.
(c) Write specifications for the following operations:
   i. An operation called *setMode* that accepts and records a value for the mode of the robot.
   ii. An operation called *getMode* that outputs the current mode of the robot.
   iii. An operation called *isIdle* that checks whether or not the robot is idle.
   iv. Modify the *setMode* operation so that the mode of a robot cannot be changed directly from broken to working.

Question 04: **[Marks: 1 + 1 + 2 * 4 = 10]**

CLO: < 4 > ***Bloom Taxonomy Level: <Applying>***

Formally specify the system in VDM-SL.

Consider a piece of software designed to keep track of damaged blocks on the surface of a disk. A disk is divided into a number of tracks and each track into a number of sectors. A block is identified, therefore, by giving both a track and sector number. Following figure gives a simplified UML specification of the DiskScanner class. DiskScanner class is specified as being a collection of Block records. A block type is track and a sector number.

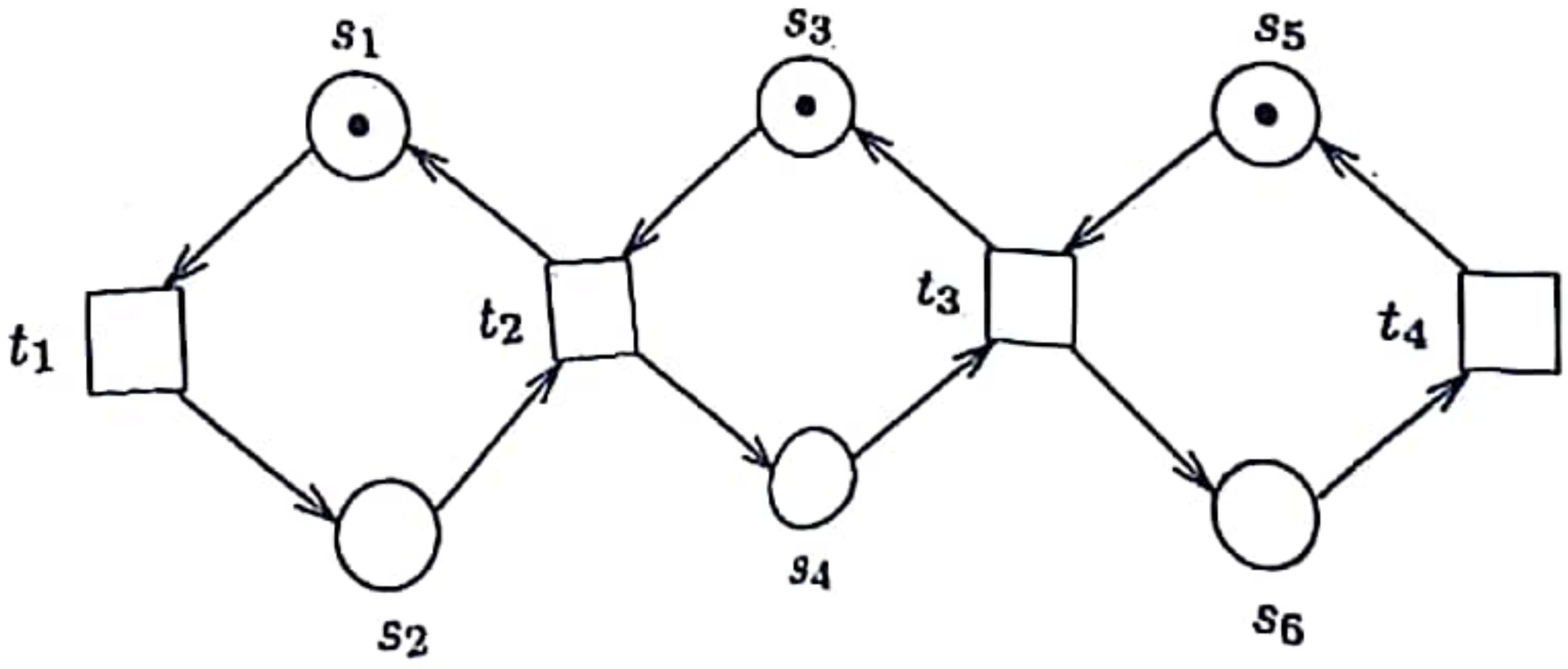


(a) Declare a type, to be used in the specification.
(b) Write the specification of the state of the system, including an initialization function.
(c) Write specifications for the four operations mentioned in the UML specification

Question 05: **[Marks: 7 + 3 = 10]**

CLO: < 3 > ***Bloom Taxonomy Level: <Applying>***

Draw a reachability Tree and Graph of the following Petri net model. Please clearly lable the markings and the transitions on each node and arc respectively.



Result ::= ok | already_known

AddBirthday

ΔBirthdayBook

name? = NAME

dateBirth? = DATE

r! = Result

name? ∉ known ⇒ (r! = ok) ∧

(birthday' = birthday ∪ {name? ↦ dateBirth?})

name? ∈ known ⇒ r! = already_known

FindBirthDay

≡ BirthdayBook

name? : NAME

date! : DATE

name? ∈ known

date! = birthday(|name?|)

AlreadyKnown

Ξ BirthdayBook

name? : NAME report! = REPORT

---

name $\in$ dom birthday

report! = already_known

(Q2)

Airport

permission : IF Aircraft

landed : IF Aircraft

---

landed ⊆ permission

#landed ≤ 20

InitAirport

▶ Airport

---

permission = {∅}

landed = {∅}

givePermission

△ Airport

plane? : Aircraft

---

plane? ∉ permission

permission' = permission ∪ {plane?}

landed' = landed

recordLanding

△ Airport

plane? : Aircraft

---

plane? ∉ landed

plane? ∈ permission

#landed < 20

landed' = landed ∪ {plane?}

permission' = permission

getPermission

≡ Airport

list! : ℙ Aircraft

list! = permission

numberWaiting

≡ Airport

num! : ℤ

num! = # (permission \ landed)

(Q3)

types

Mode : <Working> | <Idle> | <Broken>

values

state IndustrialRobot of

mode : Mode

init mk-IndustrialRobot(mode) ≜

mode = <Idle>

end

operations

setMode (newmode : Mode)

ext wr mode : Mode

pre TRUE

post mode = newmode

getMode () currMode : Mode
ext rd mode : Mode
pre TRUE
post currMode = mode

isIdle () isIdle : IB
ext rd mode : Mode
pre TRUE
post isIdle ⇔ mode = <Idle>

setMode (newMode : Mode)
ext wr mode : Mode
pre mode ≠ <Broken> ^ newMode ≠ <Working>
post mode = newMode

(84)

types

track : Z

sector : Z

Block : track → sector

values

state DiskScanner of

damagedBlocks : Block-set

init mk-DiskScanner(dB) △ dB = {}

end

operations

addBlock (track : ℤ, sector : ℤ)
ext wr damagedBlocks : Blocks-set
pre {track → sector} ∉ damagedBlocks
post damagedBlocks = ~~damagedBlocks~~ damagedBlocks ∪
{track → sector}

removeBlock (track : ℤ, sector ℤ)
ext wr damagedBlocks : Blocks-set
pre {track → sector} ∈ damagedBlocks
post damagedBlocks = ~~damagedBlocks~~ damagedBlocks \
{track, sector}

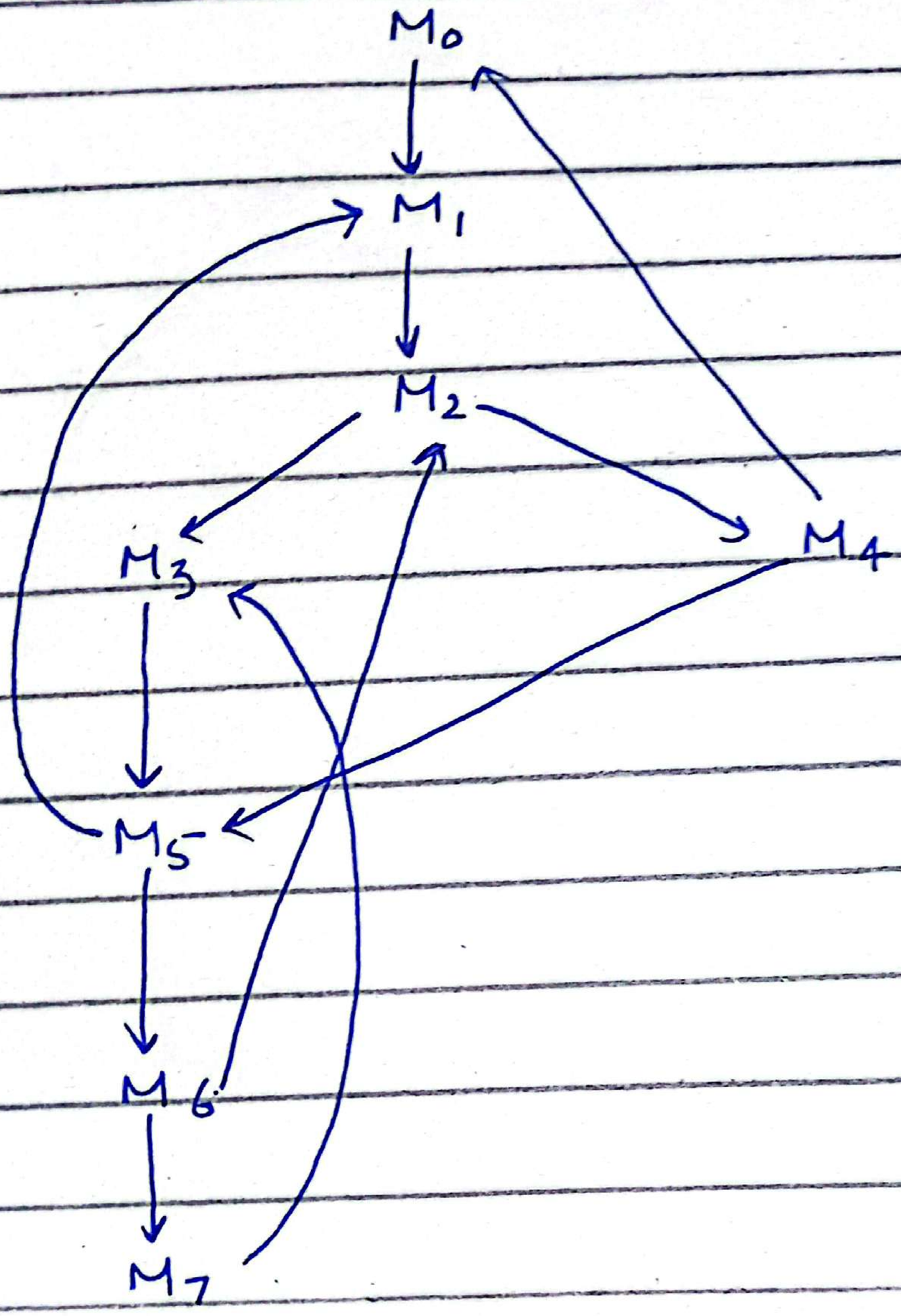
isDamageed (track : ℤ, sector ℤ) isDmg : 𝔹
ext rd dB : Blocks-set
pre ~~{track → sector} ∈ dB~~ TRUE
post isDmg ⇔ {track → sector} ∈ dB

getBadSectors ( ) sectors : ℤ-set → O clue what UML meant so returning all damaged sectors
ext rd dB : Blocks-set
pre TRUE
post sectors = ran dB

getBad

Q5)

(1,0,1,0,1,0) M0
t1
(0,1,1,0,1,0) M1
t2
(1,0,0,1,1,0) M2
t1
t3
(0,1,0,1,1,0) M3
(1,0,1,0,0,1) M4
t3
t1
(0,1,1,0,0,1) M5
(0,1,1,0,0,1) M5
(1,0,1,0,1,0) M0
t2
t4
X
X
(1,0,0,1,0,1) M6
(0,1,1,0,1,0) M1
t1
t4
X
(0,1,0,1,0,1) M7
(1,0,0,1,1,0) M2
t4
X
X
(0,1,0,1,1,0) M3
X


M0
M1
M2
M3
M4
M5
M6
M7