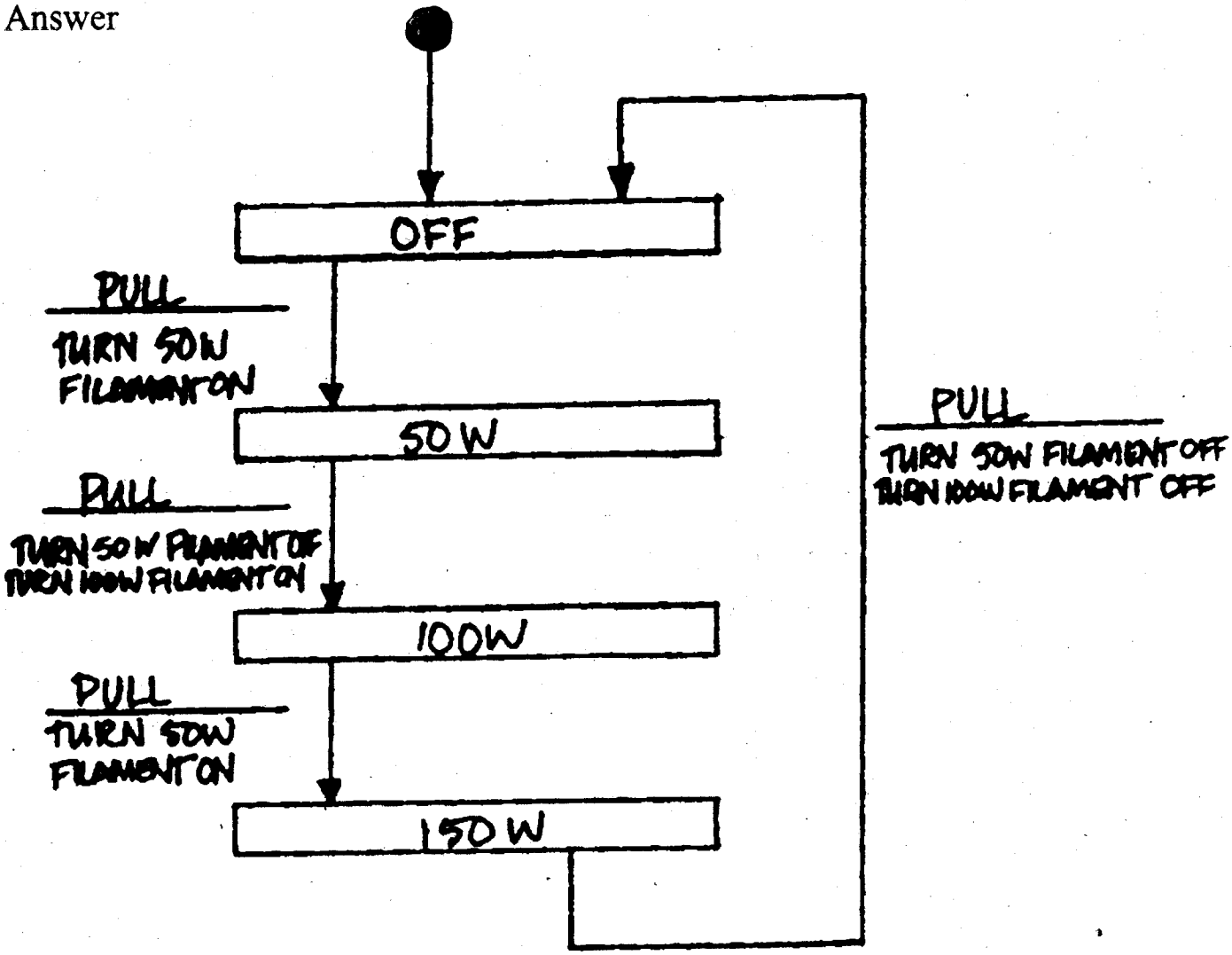
EE4001/IM2001 Software Engineering
Tutorial 6 and Sample Answer

Q1. A lamp has two filaments, one of 100 W and another of 50 W. A single switch controls the lamp so that a sequence of cord pulls will turn the lamp from being off to providing illumination at 50 W then 100 W then 150 W then back to off. Externally, there are four behaviors (off, 50 W, 100 W, and 150 W). The event flow outputs have different implications for the system depending on when they occur. For example, turning the 50 W filament on may change the lamp from off to 50 W or it may change the lamp from 100 W to 150 W. Draw a state diagram (with actions accompanied with transitions shown) to model the lamp. You may make additional assumption and state them clearly if needed.

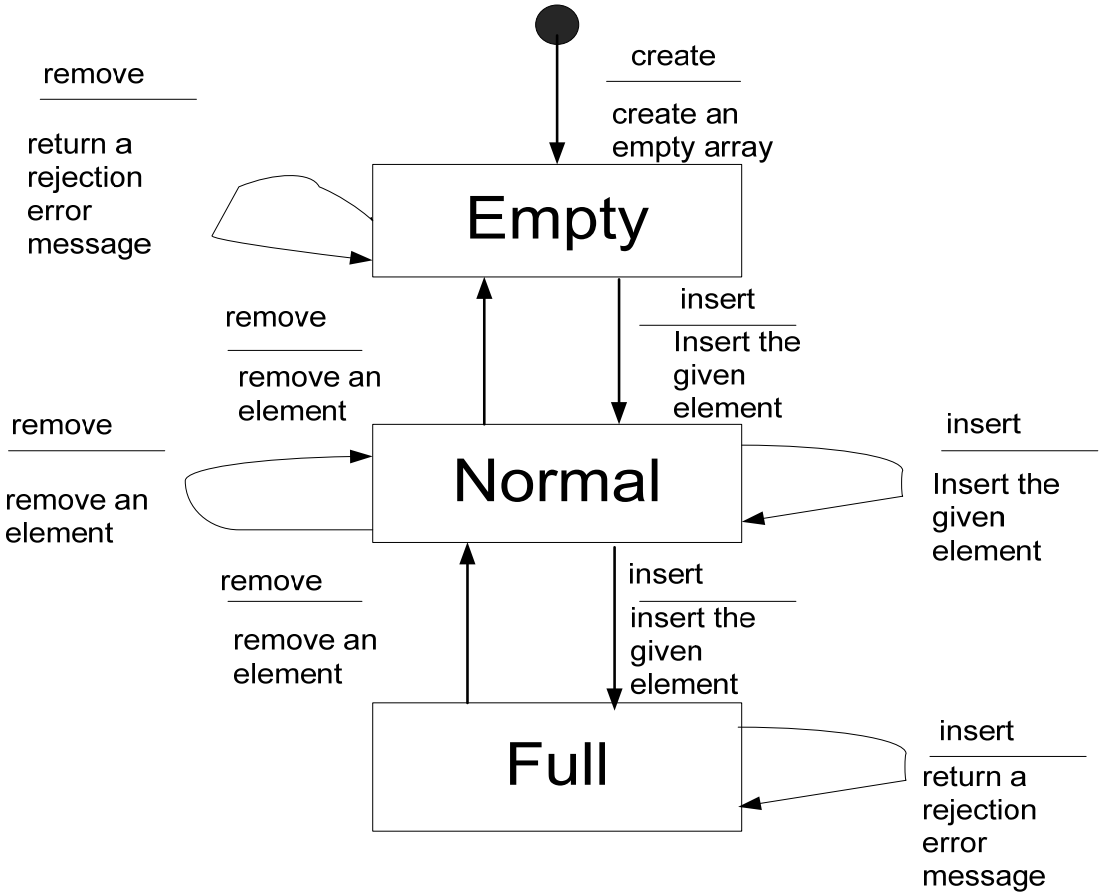
Q1 Answer



Q2. Draw a state diagram to model the following lifecycle of the data structure Array that has a maximum size:

Initially, an empty array is created. After an array is created, an element can be inserted into the array each time so long as it has not reached the maximum size. Once, the array has reached its maximum size, any attempt to insert an element into the array will return with a rejection error message. When the array has some elements, each time, an element can be removed from it. Any attempt to remove an element from an empty array will return with a rejection error message.

Q2 Answer



Remarks:
Normal: This state means the array is not empty and has not reached the maximum size

Q3. In a country, all employees must participate in a saving scheme to support their retirement. The minimum age of an employee is 18. An employee must save a certain percent of his/her monthly salary according to the following tables:

Salary/Month / **Age**	**≤ 750**	**above 750**
18 ≤ Age ≤ 35	8%	13% with salary ceiling at 5000
35 < Age ≤ 55	5%	10% with salary ceiling at 5000
Age > 55	0%	5% with salary ceiling at 5000

The amount to be saved for each month is computed by a program based on the above parameters as its input variables. Use equivalence class testing to design a test suite to test this program.

Q3 Answer:
Partition of input space:

	Salary/month ≤ 0	0 < Salary/month ≤ 750	750 < Salary/month ≤ 5000	5000 < Salary/month
Age < 18	Invalid	Invalid	Invalid	Invalid
18 ≤ Age ≤ 35	Invalid	8%	13%	13%*5000
35 < Age ≤ 55	Invalid	5%	10%	10%*5000
Age > 55	Invalid	0%	5%	5%*5000

Age = 5, Salary = -500, Invalid
Age = - 5, Salary = 500, Invalid
Age = 0, Salary = 1000, Invalid
Age = -100, Salary = 6000, Invalid

Age = 25, Salary = -1000, Invalid
Age = 18, Salary = 500, Amount to be saved = 40
Age = 20, Salary = 4500, Amount to be saved = 585
Age = 30, Salary = 90,000, Amount to be saved = 650

Age = 40, Salary = 0, Invalid
Age = 42, Salary = 300, Amount to be saved = 15
Age = 50, Salary = 3000, Amount to be saved = 300
Age = 44, Salary = 7000, Amount to be saved = 500

Age = 57, Salary = -500, Invalid
Age = 59, Salary = 750, Amount to be saved = 0
Age = 99, Salary = 5000, Amount to be saved = 250
Age = 60, Salary = 9000, Amount to be saved = 250

4. Use boundary value testing [redacted] to design a test suite to test the program specified in the question 1.

3

Q4 Answer

(Based on valid ranges, identify boundary values. A range with an open end does not have any boundary value at the end)

Boundary age:

Age = 17 (Invalid)

Age = 18

Age = 35

Age = 36

Age = 55

Age = 56

Age
Input Cond

$18 \le Age \le 35$

$35 < Age \le 55$

$Age > 55$

All

Boundary Values

18, 17, 35, 36

35, 55, 56

55

17, 18, 35, 36, 55, 56

Boundary salary/month:

Salary/month = 0 (Invalid)

Salary/month = 750

Salary/month = 760

Salary/month = 5000

Salary/month = 5100

Salary
Input Cond.

$0 < Salary \le 750$

$750 < Salary \le 5000$

$Salary > 5000$

All

Boundary Values

0, 750, 760

750, 5000, 5100

5000

0, 750, 760, 5000, 5100

Age = 18, Salary/month = 750, Amount to be saved = 60

Age = 35, Salary/month = 750, Amount to be saved = 60

Age = 36, Salary/month = 750, Amount to be saved = 37.5

Age = 55, Salary/month = 750, Amount to be saved = 37.5

Age = 56, Salary/month = 750, Amount to be saved = 0

Age = 18, Salary/month = 760, Amount to be saved = 98.8

Age = 35, Salary/month = 760, Amount to be saved = 98.8

Age = 36, Salary/month = 760, Amount to be saved = 76

Age = 55, Salary/month = 760, Amount to be saved = 76

Age = 56, Salary/month = 760, Amount to be saved = 38

Age = 18, Salary/month = 5000, Amount to be saved = 650
Age = 35, Salary/month = 5000, Amount to be saved =650
Age = 36, Salary/month = 5000, Amount to be saved = 500
Age = 55, Salary/month = 5000, Amount to be saved = 500
Age = 56, Salary/month = 5000, Amount to be saved = 250

Age = 18, Salary/month = 5100, Amount to be saved = 650
Age = 35, Salary/month = 5100, Amount to be saved =650
Age = 36, Salary/month = 5100, Amount to be saved = 500
Age = 55, Salary/month = 5100, Amount to be saved = 500
Age = 56, Salary/month = 5100, Amount to be saved = 250

Age = 17, Salary/month = 750, Invalid
Age = 17, Salary/month = 760, Invalid
Age = 17, Salary/month = 5000, Invalid
Age = 17, Salary/month = 5100, Invalid

Age = 18, Salary/month = 0, Invalid
Age = 35, Salary/month = 0, Invalid
Age = 36, Salary/month = 0, Invalid
Age = 55, Salary/month = 0, Invalid
Age = 56, Salary/month = 0, Invalid

Q5. The monthly commission to be paid to a sales executive is computed by a program according to his/her monthly sales of company's own product and agency product denoted by S and T respectively according to the following formula:

$$\begin{aligned} \text{Monthly Commission} &= S*0.01 + T*0.01 && \text{if } S \geq T \\ &= S*0.01 + T*0.005 && \text{if } S < T \end{aligned}$$

This program accepts S and T as its input from users. Both S and T must be nonnegative. That is $S \geq 0$, $T \geq 0$.
Use cause-effect testing technique to design a test suite to test this program, through constructing a decision table with minimal number of basic conditions.

Q5 Answer

We have the following basic conditions and actions extracted from the specification:

Basic Conditions:	**Actions:**
(1) $S \geq 0$	(11) Monthly Commission = S*0.01 + T*0.01
(2) $T \geq 0$	(12) Monthly Commission = S*0.01 + T*0.005
(3) $S \geq T$	
(4) $S < T$	
(5) $S < 0$	
(6) $T < 0$	

When we check these basic conditions in the order shown one by one and include it in the decision table only if it cannot be derived from those that have been included, after we include the first three, we find that:

($S<T$) can be derived from ($S \geq T$)
($S < 0$) can be derived from ($S \geq 0$)
($T < 0$) can be derived from ($T \geq 0$)

Hence, we eliminate condition (4), (5) and (6) and only include the first three conditions in the decision table.

We construct the following decision table based on the minimal set of basic conditions identified:

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Conditions						
$S \geq 0$ (1)	T	T	T	F	F	F
$T \geq 0$ (2)	T	T	F	T	F	F
$S \geq T$ (3)	T	F	T	F	T	F
Actions						
Monthly Commission = S*0.01 + T*0.01 (11)	T	F	I	I	I	I
Monthly Commission = S*0.01 + T*0.005 (12)	F	T	I	I	I	I

I: Invalid Case

Note that for one F case, we do not include ($S \geq 0$) = F, ($T \geq 0$) = T and ($S \geq T$) = T as a rule. For two F case, we do not include ($S \geq 0$) = T, ($T \geq 0$) = F and ($S \geq T$) = F as a rule. The reason is that these two combinations of conditions values are not possible.

From the decision table, we design a test case for each rule as follows:

Rule 1: S = 500,000, T = 500,000, Monthly Commission = 500,000 * 0.01 + 500,000 * 0.01 = 10,000.

Rule 2: S = 500,000, T = 1,000,000, Monthly Commission = 500,000 * 0.01 + 1,000,000 * 0.005 = 10,000.

Rule 3: S = 500,000, T = -500,000, Invalid Case.

Rule 4: S = -500,000, T = 500,000, Invalid Case.

Rule 5: S = -500,000, T = -1,00,000, Invalid Case.

Rule 6: S = -1,000,000, T = -500,000, Invalid Case.