Question 1: A mobile phone service provider uses a program that computes the monthly bill of customers as follows:

Minimum Rs 300 for up to 120 calls

Plus, Re 1 per call for the next 70 calls

Plus Rs 0.80 per call for the next 50 calls

Plus Rs 0.40 per call for any call beyond 220 calls.

Design test cases for this program using equivalence class testing technique.

Solution:

Let variable A for number of phone calls.

First, we partition the domain of input as valid input values and invalid values, getting the following classes:

I_1= { <A> : $1 \le A \le 120$}

I_2= { <A> : $121 \le A \le 190$}

I_3= { <A> : $191 \le A \le 240$}

I_4= { <A> : $A \ge 220$}

I_5= { <A> : $A < 0$}

Now the test cases can be designed from the above derived classes, taking one test case from each class such that the test case covers maximum valid input classes, and separate test cases for each invalid class.

The test cases are shown below:

Test Case Id	Input Value A	Expected Output	Classes covered by the test case
1	90	300 Rs	I_1
2	145	325 Rs	I_2
3	210	372 Rs	I_3
4	250	352	I_4
5	-2	Invalid Input	I_5

Question 2:

A program reads players' records with the following detail and prints a team-wise list containing player name with their batting average:

Player name (max. 30 characters)

Team name (max. 20 characters)

Batting average

Design test cases for this program using BVC, robust testing, and worst-case testing methods

Solution:

(a) Test cases using BVC Since there are three variables player name, team name and batting average but there is no boundary for batting average value so the total number variable 2 and total number of test cases will be 4n + 1 = 9.

The set of boundaries values is shown below.

For player name

Min value = 1 character

Min^{+} value = 2 character

Max value = 30 character

Max^{-} value = 29 character

Nominal value = 15 character

For team name

Min value = 1 character

Min^{+} value = 2 character

Max value = 20 character

Max^{-} value = 19 character

Nominal value = 10 character

Test Case ID	Player name	Length (P.name)	Team Name	Length (T.name)	Batting Average	Expected Output
1	r	1	ridersrider	10	44	Print(player and team name with batting average)
2	Ra	2	ridersrider	10	44	Print(player and team name with batting average)
3	Abababababababababababababababab	30	ridersrider	10	34	Print(player and team name with batting average)
4	Ababababababababababababababa	29	ridersrider	10	44	Print(player and team name with batting average)
5	ababbaababababa	15	R	1	5	Print(player and team name with batting average)
6	anababababababb	15	Rr	2	44	Print(player and team name with batting average)
7	ragahahahahaha	14	ridersriderridersrider	20	54	Print(player and team name with batting average)

8	ababbaababababau	16	ridersrider ridersride	19	33	Print(player and team name with batting average)
9	ababbaababababa	15	ridersrider	10	22	Print(player and team name with batting average)

(b) **Test cases using robust testing** Since there are two variables player name and team name, the total number of test cases will be 6n + 1 = 13.

The set of boundary values is shown below:

For player name

Min value = 1 character

Min$^+$ value = 2 character

Min$^-$ value = 0 or Null character

Max value = 30 character

Max$^+$ value = 31 character

Max$^-$ value = 29 character

Nominal value = 15 character

For team name

Min value = 1 character

Min$^+$ value = 2 character

Min$^-$ value = 0 or Null character

Max value = 20 character

Max$^+$ value = 21 character

Max$^-$ value = 19 character

Nominal value = 10 character

Test Case ID	Player name	Length (P.name)	Team Name	Length (T.name)	Batting Average	Expected Output
1	r	1	ridersrider	10	44	Print(player and team name with batting average)

2	Ra	2	ridersrider	10	44	Print(player and team name with batting average)
3	Abababababababab ababababababab	30	ridersrider	10	34	Print(player and team name with batting average)
4	Abababababababab ababababababa	29	ridersrider	10	44	Print(player and team name with batting average)
5	ababbaababababa	15	R	1	5	Print(player and team name with batting average)
6	anababababababb	15	Rr	2	44	Print(player and team name with batting average)
7	ragahahahahaha	14	ridersriderridersrider	20	54	Print(player and team name with batting average)
8	ababbaababababau	16	ridersrider ridersride	19	33	Print(player and team name with batting average)
9	ababbaababababa	15	ridersrider	10	22	Print(player and team name with batting average)
10		0	ridersrider	10	23	Invalid Input
11	ababbaababababa	15		0	23	Invalid Input
12	Abababababababab Ababababababab1	31	ridersrider	10	23	Invalid Input
13	ababbaababababa	15	ridersriderridersridere	21	45	Invalid Input

C) Test cases using worst-case testing Since there are two variables, player name and team name, the total number of test cases will be

5^n= 25.

Test Case ID	Player name	Length (P.name)	Team Name	Length (T.name)	Batting Average	Expected Output
1	r	1	ridersrider	10	44	Print(player and team name with batting average)
2	Ra	2	ridersrider	10	44	Print(player and team name with batting average)
3	Abababababababab ababababababab	30	ridersrider	10	34	Print(player and team name with batting average)

4	Ababababababababababababababa	29	ridersrider	10	44	Print(player and team name with batting average)
5	ababbaababababa	15	R	1	5	Print(player and team name with batting average)
6	anababababababb	15	Rr	2	44	Print(player and team name with batting average)
7	ragahahahahaha	14	ridersriderridersrider	20	54	Print (player and team name with batting average)
8	ababbaababababau	16	ridersrider ridersride	19	33	Print (player and team name with batting average)
9	ababbaababababa	15	ridersrider	10	22	Print (player and team name with batting average)
-----	------	------	-------------	-----	-----	------
-----	------	------	-------------	-----	-----	------

Question 3:

A university is admitting students in a professional course subject to the following conditions:

(a) Marks in Java ≥ 70

(b) Marks in C++ ≥ 60

(c) Marks in OOAD ≥ 60

(d) Total in all three subjects ≥ 220 OR Total in Java and C++ ≥ 150

If the aggregate mark of an eligible candidate is more than 240, he will be eligible for scholarship course, otherwise he will be eligible for normal course. The program reads the marks in the three subjects and generates the following outputs:

(i) Not eligible

(ii) Eligible for scholarship course

(iii) Eligible for normal course

Design test cases for this program using equivalence class testing technique.

Solution:

Let variable J for marks in java, variable J for marks in Java, C for marks in C++, O for marks in OOAD.

First, we partition the domain of input as valid input values and invalid values, getting the following classes:

I_1= { <J> : J ≥ 70}

I_2= { <C> : J ≥ 60}

I_3= { <O> : O ≥ 60}

I_4= { <J,C,O> : J +C+D ≥ 220}

I_5= { <J,C> : J +C ≥ 150}

I_6= { <J,C,O> : J +C+D ≥ 240}

I_7= { <J> : J < 70}

I_8= { <C> : C < 60}

I_9= { <O> : O < 60}

I_{10}= { <J,C,O> : J +C+D < 220}

I_{11}= { <J,C> : J +C < 150}

I_{12}= { <J,C,O> : J +C+D <240}

Test Case Id	Java(J)	C++(C)	OOAD(O)	Expected Output	Classes covered by the test case
1	90	60	60	Eligible for normal course	I_1 ,I_2 ,I_3 ,I_5 ,I_{10} , I_{12}
2	90	60	80	Eligible for normal course	I_1 ,I_2 ,I_3 ,I_4 , I_{12}
3	90	80	90	Eligible for scholarship course	I_1 ,I_2 ,I_3 ,I_5 ,I_4,I_6
4	50	80	90	Not eligible	I_7 ,I_2 ,I_3 ,I_{11} ,I_{10} , I_{12}
5	90	50	90	Not eligible	I_1 ,I_8 ,I_3 ,I_{11} ,I_{10} , I_{12}
6	80	80	50	Not eligible	I_1 ,I_2 ,I_9 ,I_{11} ,I_{10} , I_{12}
7	70	60	60	Not eligible	I_1 ,I_2 ,I_3 , I_{11} ,I_{10} , I_{12}