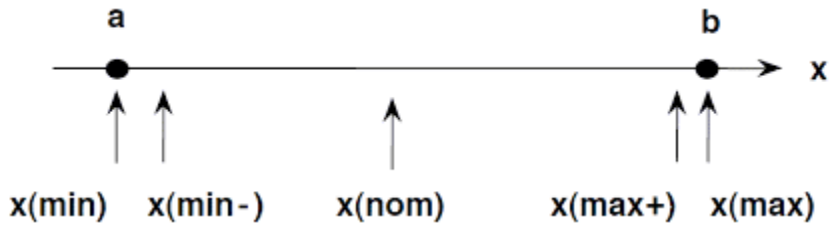
BOUNDARY VALUE TESTING

- Boundary value analysis is a **black-box testing technique** that helps test whether an application's output falls within the acceptable range for all input values. It focuses on testing edge or boundary conditions, including extreme and invalid values. It identifies errors or imperfections in input parameter boundary conditions. BVA is a key testing technique that helps identify and fix bugs early in the software development lifecycle (SDLC). This article describes the basics of marginal analysis, its benefits, and how to perform marginal analysis tests effectively.

Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.

- So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called "boundary testing".
- The basic idea in normal boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum



- In Boundary Testing, Equivalence Class Partitioning plays a good role
- Boundary Testing comes after the Equivalence Class Partitioning.

- **Real-life example of boundary value analysis**
- **Example 1**
- Consider a software application that requires users to enter the age. Applications may contain specific minimum and maximum age restrictions. The restriction is to enter the age between 18 to 30 years. If the user enters values above 30 or below 18, the application may not work as expected. In this scenario, you can use BVA to test your application by choosing values equal to, above, or below the age limit.

Invalid case	**Valid case**	**Invalid case**
11.12.13.,14,15,16,17	21,22,23,24,25,26,27,28,29	31,32,33,34,35,36,37,38,39

Equivalence class testing (Equivalence class Partitioning) is a black-box testing technique used in software testing as a major step in the Software development life cycle (SDLC). This testing technique is better than many of the testing techniques like boundary value analysis, worst case testing, robust case testing and many more in terms of time consumption and terms of precision of the test cases. Since testing is done to identify possible risks, equivalence class testing performs better than the other techniques as the test cases generated using it are logically identified with partitions in between to create different input and output classes. This can be shown from the next-date problem which is stated below:
Given a day in the format of day-month-year, you need to find the next date for the given date. Perform boundary value analysis and equivalence-class testing for this.

Conditions :

D: 1<Day<31

M: 1<Month<12

Y: 1800 <Year <2048

No. of test Cases (n = no. of variables) = **4n+1** = 4*3 +1 =13

Test Cases:

Test Case ID	Day	Month	Year	Expected Output
1	1	6	2000	2-6-2000
2	2	6	2000	3-6-2000
3	15	6	2000	16-6-2000
4	30	6	2000	1-7-2000
5	31	6	2000	Invalid Date
6	15	1	2000	16-1-2000
7	15	2	2000	16-2-2000
8	15	11	2000	16-11-2000
9	15	12	2000	16-12-2000
10	15	6	1800	16-6-1800
11	15	6	1801	16-6-1801
12	15	6	2047	16-6-2047
13	15	6	2048	16-6-2048

Equivalence Class Testing:

Input classes:

Day:

D1: day between 1 to 28

D2: 29

D3: 30

D4: 31

Month:

M1: Month has 30 days

M2: Month has 31 days

M3: Month is February

Year:

Y1: Year is a leap year

Y2: Year is a normal year

Output Classes:

Increment Day

Reset Day and Increment Month

Increment Year

Invalid Date **Strong Normal Equivalence Class Test Cases:**

TEST CASE ID	DAY	MONTH	YEAR	EXPECTED OUTPUT
1	D1	M1	Y1	Increment day
2	D1	M1	Y2	Increment day
3	D1	M2	Y1	Increment day
4	D1	M2	Y2	Increment day
5	D1	M3	Y1	Increment day or Reset day and Increment month
6	D1	M3	Y2	Increment day or Reset day and Increment month
7	D2	M1	Y1	Increment day
8	D2	M1	Y2	Increment day
9	D2	M2	Y1	Increment day
10	D2	M2	Y2	Increment day
11	D2	M3	Y1	Reset day and Increment month
12	D2	M3	Y2	Invalid Date
13	D3	M1	Y1	Reset day and Increment month
14	D3	M1	Y2	Reset day and Increment month
15	D3	M2	Y1	Increment day
16	D3	M2	Y2	Increment day
17	D3	M3	Y1	Invalid Date
18	D3	M3	Y2	Invalid Date
19	D4	M1	Y1	Invalid Date
20	D4	M1	Y2	Invalid Date
21	D4	M2	Y1	Reset day and Increment month
22	D4	M2	Y2	Reset day and Increment month
23	D4	M3	Y1	Invalid Date
24	D4	M3	Y2	Invalid Date

Test Cases:

Test Case ID	Day	Month	Year	Expected Output
E1	15	4	2004	16-4-2004
E2	15	4	2003	16-4-2003
E3	15	1	2004	16-1-2004
E4	15	1	2003	16-1-2003
E5	15	2	2004	16-2-2004

E6	15	2	2003	16-2-2003
E7	29	4	2004	30-4-2004
E8	29	4	2003	30-4-2003
E9	29	1	2004	30-1-2004
E10	29	1	2003	30-1-2003
E11	29	2	2004	1-3-2004
E12	29	2	2003	Invalid Date
E13	30	4	2004	1-5-2004
E14	30	4	2003	1-5-2003
E15	30	1	2004	31-1-2004
E16	30	1	2003	31-1-2003
E17	30	2	2004	Invalid Date
E18	30	2	2003	Invalid Date
E19	31	4	2004	Invalid Date
E20	31	4	2003	Invalid Date

E21	31	1	2004	1-2-2004
E22	31	1	2003	1-5-2003
E23	31	2	2004	Invalid Date
E24	31	2	2003	Invalid Date

So from this problem it is clearly seen that equivalence class testing clearly checks for many cases that boundary value did not considered like that of February which has 28-29 days, leap year which lead to variation in number of days in February and many more.

Hence the above example proves that equivalence partitioning generates more efficient test cases that should be considered during risk assessment.