Lab Assignment : 09

- Test Case generation by Equivalence Partitioning Technique for Q1:
  - Neel Makadiya (201801190)
  - Rishiraj Meena (201801096)

- Test Case generation by Boundary Value Analysis for Q1 :
  - Gunank Garg (201801078)
  - Harsh Vora (201801136)

- Program of 1st problem in C++
  - Yagnik Sakhiya (201801175)

- Test Case generation by Equivalence Partitioning Technique for Q2:
  - Rutwa Rami (201801205)
  - Ishwa Bhatt (201801445)

- Test Case generation by Boundary Value Analysis for Q2 :
  - Tirth Patel (201801126)
  - Jay Shah (201801133)
  - Siddhraj Parmar (201801466)

Question 01 :
Equivalence Partitioning Technique :

1<= Day <= 31 && 1<=Month<=12 && 1900<=year<=2015

- Any Day < 1 => Invalid
  Day between 1 and 31 (Including both 1 and 31) => Valid
  Any Day > 31 => Invalid

- Any Month < 1 => Invalid
  Month Between 1 and 12 (Including both 1 and 12) => Valid
  Any Month > 12 => Invalid

- Any Year < 1900 => Invalid
  Year Between 1900 and 2015 (Including Both) => Valid
  Year > 2015 => Invalid

➔ If any one field of this three is Invalid then whole Date is invalid.

➔ Set of Value for Date = {0,15,40}
➔ Set of Value for Month = {0,6,15}
➔ Set of value for Year = {1800,2000,2020}
➔ Total possible combination = 3*3*3 = 27

Test Case and Expected Result :

Test Case Number	Day	Month	Year	Result
1	0	0	1800	Invalid
2	0	0	2000	Invalid
3	0	0	2020	Invalid
4	0	6	1800	Invalid
5	0	6	2000	Invalid
6	0	6	2020	Invalid
7	0	15	1800	Invalid

8	0	15	2000	Invalid
9	0	15	2020	Invalid
10	15	0	1800	Invalid
11	15	0	2000	Invalid
12	15	0	2020	Invalid
13	15	6	1800	Invalid
14	15	6	2000	14-6-2000
15	15	6	2020	Invalid
16	15	15	1800	Invalid
17	15	15	2000	Invalid
18	15	15	2020	Invalid
19	40	0	1800	Invalid
20	40	0	2000	Invalid
21	40	0	2020	Invalid
22	40	6	1800	Invalid
23	40	6	2000	Invalid
24	40	6	2020	Invalid
25	40	15	1800	Invalid
26	40	15	2000	Invalid
27	40	15	2020	Invalid

Considering the Date validation (Further validation):

- Day :

D1 : Day between 1 and 28 (Including both 1 and 28)
D2 : Day = 29
D3 : Day = 30
D4 : Day = 31

- Month :

M2 : {1,3,5,7,8,10,12}	(Month which has 31 days)
M3 : {4,6,9,11}	(Month which has 30 days)
M4 : {2}	(Month which has 28/29 days)

- Year :

Y1 : Given year is leap year
Y2 : Given Year is not leap Year

➔ Set of value for Day ={15,29,30,31}
➔ Set of value for Month = {7,9,2}
➔ Set of value for year = {2000, 2011}

Test Case and Expected Result :

Test Case Number	Day	Month	Year	Expected Result
1	15	7	2000	14-7-2000
2	15	7	2011	14-7-2011
3	15	9	2000	14-9-2000
4	15	9	2011	14-9-2011
5	15	2	2000	14-2-2000
6	15	2	2011	14-2-2011
7	29	7	2000	28-7-2000
8	29	7	2011	28-7-2011

9	29	9	2000	28-9-2011
10	29	9	2011	28-9-2011
11	29	2	2000	28-2-2000
12	29	2	2011	Invalid
13	30	7	2000	29-7-2000
14	30	7	2011	29-7-2011
15	30	9	2000	29-9-2000
16	30	9	2011	29-9-2011
17	30	2	2000	Invalid
18	30	2	2011	Invalid
19	31	7	2000	30-7-2000
20	31	7	2011	30-7-2011
21	31	9	2000	Invalid
22	31	9	2011	Invalid
23	31	2	2000	Invalid
24	31	2	2011	Invalid

Boundary Test Cases:

As per boundary cases format value set ({ min , min+1 , max-1 , max }) day’s, month’s and year’s sets are as follow

Day : {1,2,15,30,31}
Month : {1,2,6,11,12}
Year : {1900,1901,2000,2014,2015}

And we add 29 to the day's set for checking leap year constraints.

Test Case ID	Day	Month	Year	Expected Output
1	1	6	2000	31-5-2000
2	2	6	2000	1-6-2000
3	30	6	2000	29-6-2000
4	31	6	2000	Invalid input
5	15	1	2000	14-1-2000
6	15	2	2000	14-2-2000
7	15	11	2000	14-11-2000
8	15	12	2000	14-12-2000
9	15	6	1900	14-6-1900
10	15	6	1901	14-6-1901
11	15	6	2014	14-6-2014
12	15	6	2015	14-6-2015
13	29	2	2010	Invalid input
14	29	2	2012	28-2-2012

Program in C++ :

```
#include <bits/stdc++.h>
using namespace std;

bool checkYear(int year)
{
    if (year % 400 == 0)
        return true;

    if (year % 100 == 0)
        return false;

    if (year % 4 == 0)
        return true;

    return false;
}

int main()
{
    int date, month, year;

    cin >> date >> month >> year;

    int day_in_month[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30,
31, 30, 31};

    bool lear_year = checkYear(year);

    if (lear_year)   day_in_month[2]++;

    if (date < 1 || date > 31) {
        cout << "Invalid Input";
    }

    else if (month < 1 || month > 12) {
        cout << "Invald Input";
    }
    else if (year < 1900 || year > 2015) {
        cout << "Invalid Input";
    }

    else if (date > day_in_month[month]) {
        cout << "Invalid Input";
    }

    else {
        int pre_date = date - 1, pre_month = month, pre_year = year;

        if (pre_date == 0) {
            pre_month--;

            if (pre_month == 0) {
                pre_month = 12;

                pre_year--;
            }

            pre_date = day_in_month[pre_month];

        }
        cout << pre_date << " - " << pre_month << " - " << pre_year
<< "\n";
    }
}
```

Question : 02

Constraints:
ID: 00000 - 99999
Quantity : 1 - 99
Cart total : Maximum $999.99

Equivalence class : -

1) 00000<= ItemID <= 99999, 0 <= Quantity <= 99, 0 <= Cart total <= $999.99 (valid partition)
2) ItemID < 00000 (Invalid partition, for any quantity)
3) ItemID > 99999 (Invalid partition, for any quantity)
4) Quantity < 0 (Invalid partition, for any ItemID)
5) Quantity > 99 (Invalid partition, for any ItemID)
6) Cart Total > $999.99 (Invalid partition, for any ItemID or Quantity)

Test Case	Inputs	Outputs
1	ItemId: 32014 Quantity: 12	Valid, Cart total will be displayed (here, cart total is less than or equal to $999.99)
2	ItemID: -16	Invalid
3	ItemId: 100045	Invalid
4	ItemID: 12345 (any valid id) Quantity: -54	Invalid
5	ItemID: 23456 (any valid id) Quantity: 120	Invalid
6	ItemID: 70023 (any valid id) Quantity: 90 (Item Price: $200)	Invalid
7	ItemID: 12345 (any valid id) Quantity: 0	Item will be removed if the ItemID was added previously in the list
8	ItemID: 12345 (any valid id)	Invalid, if the ItemID is not in the

	Quantity: 0	list

Boundary Value Analysis : -

For ID:

1) ID = 00000
2) ID = 00001
3) ID = 32456
4) ID = 99998
5) ID = 99999

For Quantity :

1) Quantity = 1
2) Quantity = 0
3) Quantity = 98
4) Quantity = 99

Max-Cart Total:

1) Cart_total= $0
2) Cart_total= $1.00
3) Cart_total= $998.99
4) Cart_total = $999.99

Test Cases :

Test Case	Input	Output
ID < 00000	ID = -00001	Error
ID > 99999	ID = 100000	Error
Valid ID	ID = 25534	Add to Cart

Quantity < 0	Quantity = -1	Error
Quantity = 0	ID = 65423 (present in the cart)	Remove item from cart with given ID
Quantity = 0	ID = 32456 (not present in cart)	Error[Item with given ID is not not added previously]
Valid Quantity	Quantity = 56	Add item to cart
Quantity > 99	Quantity = 100	Error
Valid Cart total	ID = 13289 , Quantity = 47	Cart_Total = $ 560
Invalid Cart total	ID = 98546 , Quantity = 34	Cart_Total = $ 3561 (Error because Cart_total>$999.99)