IT314 - Software Engineering
Lab Session-Specification-based Test Case Generation
Group 7

Group Members

- 201801242-PATEL PRIYA HARSHADBHAI
- 201801083-NANAVATI SWAPNIL DIPANSHU
- 201801115-JIMIL DIPESHKUMAR
- 201801227-PANCHAL KRUTARTH BHAVESHBHAI
- 201801088-SOLANKI ABHISHEK PRIYAKANT
- 201801173-MANGUKIYA ARSHIT DIPAKBHAI
- 201801167-SHAH DHANVI SWETAL
- 201801045-TANAYA ALPESHBHAI SHAH
- 201801055-KHATRI RAHUL NARAIN
- 201801031-SRISHTI KALRA

Section A

Question 1 : Consider a program for determining the previous date. Its input is triple of day, month and year with the following ranges 1 <= month <= 12, 1 <= day <= 31, 1900 <= year <= 2015.The possible output dates would be previous date or invalid date. Design the equivalence class test cases?

Given,

(a) 1 <= Days <= 31
(b) 1 <= Month <= 12
(c) 1900 <= Year <= 2015

Equivalence Classes:

Days:

1) Invalid Partition - Days more than 31
2) Valid Partition - Days between 1 to 31(Both Inclusive)
3) Inavlid Partition - Day less than 1

Month:

4) Invalid Partition - Month less than 1
5) Valid Partition - Month between 1 to 12 (Both Inclusive)
6) Invalid Partition - Month more than 12

Year:

7) Invalid Partition - Year less than 1900
8) Valid Partition - Year between 1900 to 2015 (Both Inclusive)
9) Invalid Partition - Year more than 2015

Total number of equivalence cases which can be constructed (both valid and invalid) from above 9 conditions are 3*3*3 = 27. These are listed down:

1) **1 <= Days <= 31, 1 <= Month <= 12, 1900 <= Year <= 2015 - Valid**
2) 1 <= Days <= 31, 1 <= Month <= 12, Year < 1900 - Invalid
3) 1 <= Days <= 31, 1 <= Month <= 12, Year > 2015 - Invalid
4) 1 <= Days <= 31, Month <1, 1900 <= Year <= 2015 - Invalid
5) 1 <= Days <= 31, Month < 1, Year < 1900 - Invalid
6) 1 <= Days <= 31, Month < 1, Year > 2015 - Invalid

7) 1 <= Days <= 31, Month > 12, 1900 <= Year <= 2015 - Invalid
8) 1 <= Days <= 31, Month > 12, Year < 1900 - Invalid
9) 1 <= Days <= 31, Month > 12, Year > 2015 - Invalid
10) Days < 1, 1 <= Month <= 12, 1900 <= Year <= 2015 - Invalid
11) Days < 1, 1 <= Month <= 12, Year < 1900 - Invalid
12) Days < 1, 1 <= Month <= 12, Year > 2015 - Invalid
13) Days < 1, Month < 1, 1900 <= Year <= 2015 - Invalid
14) Days < 1, Month < 1, Year < 1900 - Invalid
15) Days < 1, Month < 1, Year > 2015 - Invalid
16) Days < 1, Month > 12, 1900 <= Year <= 2015 - Invalid
17) Days < 1, Month > 12, Year < 1900 - Invalid
18) Days < 1, Month > 12, Year > 2015 - Invalid
19) Days > 31, 1 <= Month <= 12, 1900 <= Year <= 2015 - Invalid
20) Days > 31, 1 <= Month <= 12, Year < 1900 - Invalid
21) Days > 31, 1 <= Month <= 12, Year > 2015 - Invalid
22) Days > 31, Month < 1, 1900 <= Year <= 2015 - Invalid
23) Days > 31, Month < 1, Year < 1900 - Invalid
24) Days > 31, Month < 1, Year > 2015 - Invalid
25) Days > 31, Month > 12, 1900 <= Year <= 2015 - Invalid
26) Days > 31, Month > 12, Year < 1900 - Invalid
27) Days > 31, Month > 12, Year > 2015 - Invalid

Pseudocode to find previous date:

- If entered day=1 && month=1 then set day=31, month=12 and decrement year by 1.
- If entered day=31 then set day=1 and decrement month by 1.
- In other cases, decrement day by 1.
- If the previous date, month, year do not belong to a valid equivalence class (i.e, 1<=Days<=31, 1<=Month<=12, 1900<=Year<=2015), then the entered date is invalid.

1) Test Cases :

(Done with equivalent partitioning testing unless boundary analysis mentioned)

Belongs to equivalence class -	**Day**	**Month**	**Year**	**Validity status**
1	23	2	2001	Valid
2 (Boundary Analysis)	31	4	**1899**	Invalid
3 (Boundary	14	12	**2016**	Invalid

Analysis)				
4 (Boundary Analysis)	9	**0**	2013	Invalid
5	15	**-2**	**1865**	Invalid
6	27	**-4**	**2020**	Invalid
7	4	**14**	1993	Invalid
8	26	**22**	**1862**	Invalid
9	22	**19**	**2018**	Invalid
10 (Boundary Analysis)	**0**	7	**1903**	Invalid
11	**-2**	11	**1882**	Invalid
12	**-6**	5	**2019**	Invalid
13 (Boundary Analysis)	**-21**	**0**	**1915**	Invalid
14	**-2**	**-6**	**1775**	Invalid
15	**-8**	**-8**	**2016**	Invalid
16	**-4**	**18**	2002	Invalid
17	**-13**	**17**	**1723**	Invalid
18 (Boundary Analysis)	**-3**	**13**	**2016**	Invalid
19 (Boundary Analysis)	**36**	8	2015	Invalid
20	39	5	**1836**	Invalid
21	**32**	2	2021	Invalid
22 (Boundary Analysis)	**36**	**0**	1900	Invalid
23 (Boundary Analysis)	**42**	**0**	**1876**	Invalid
24	**34**	**-1**	**2019**	Invalid
25	**33**	**18**	2010	Invalid

26	**35**	**13**	**1887**	Invalid
27	**32**	**14**	**2018**	Invalid

*the values in bold signify the reason that it belongs to Invalid class.

2) Executable code C++ :

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

int main()
{
        int day, month, year;
        cin >> day >> month >> year;

  if((day<1 or day>31) or (month<1 or month>12) or (year<1900 or year>2015)){
        cout << "Invalid";
        return 0;
  }

  else if(day==1){
        day = 31;

        if(month == 1){
                month = 12;
                year -= 1;
        }

        else{
                month -= 1;
        }
  }

  else{
        day -= 1;
  }

  string mon;

  if(month == 1) mon = "january";
  if(month == 2) mon = "february";
  if(month == 3) mon = "march";
  if(month == 4) mon = "april";
  if(month == 5) mon = "may";
  if(month == 6) mon = "june";
  if(month == 7) mon = "july";
  if(month == 8) mon = "august";
```

```
    if(month == 9) mon = "september";
    if(month == 10) mon = "october";
    if(month == 11) mon = "november";
    if(month == 12) mon = "december";

    cout << day << " " << mon << ", " << year << "\n";
}
```

Question 2 : You are testing an e-commerce system that sells products like caps and jackets. The problem is to create functional tests using boundary-value analysis and equivalence class partitioning techniques for the webpage is shown below.

The system accepts a five-digit numeric item ID number from 00000 to 99999. The system accepts a quantity to be ordered, from 1 to 99. If the user enters a previously ordered item ID and a 0 quantity to be ordered, that item is removed from the shopping cart. Based on these inputs, the system retrieves the item price, calculates the item total (quantity times item price), and adds the item total to the cart total. Due to limits on credit card orders that can be processed, the maximum cart total is $999.99.

Given,

a) Item ID - 00000 to 99999
b) Quantity - 1 to 99
c) Cart-Total - Less than or Equal to $999.99

Equivalence Classes:

For Item ID :

1) 00000 <= Item ID <= 99999
2) Item ID > 99999
3) Item ID < 00000

For Quantity :

1) 1 <= Quantity <= 99
2) Quantity < 1
3) Quantity > 99

For Cart total:

1) 0 <= Cart total <= $999.99
2) Cart total > $999.99

Total number of equivalence cases which can be constructed (both valid and invalid) are 3*3*2 = 18. These are listed down :

1) **00000 <= Item ID <= 99999, 1 <= Quantity <= 99, 0 <= Cart total <= $999.99 - Valid**
2) 00000 <= Item ID <= 99999, 1 <= Quantity <= 99, Cart total > $999.99 - Invalid
3) 00000 <= Item ID <= 99999, Quantity < 1, 0 <= Cart total <= $999.99 - Invalid
4) 00000 <= Item ID <= 99999, Quantity < 1, Cart total > $999.99 - Invalid
5) 00000 <= Item ID <= 99999, Quantity > 99, 0 <= Cart total <= $999.99 - Invalid
6) 00000 <= Item ID <= 99999, Quantity > 99, Cart total > $999.99 - Invalid
7) Item ID > 99999, 1 <= Quantity <= 99, 0 <= Cart total <= $999.99 - Invalid
8) Item ID > 99999, 1 <= Quantity <= 99, Cart total > $999.99 - Invalid
9) Item ID > 99999, Quantity < 1, 0 <= Cart total <= $999.99 - Invalid
10) Item ID > 99999, Quantity < 1, Cart total > $999.99 - Invalid
11) Item ID > 99999, Quantity > 99, 0 <= Cart total <= $999.99 - Invalid
12) Item ID > 99999, Quantity > 99,Cart total > $999.99 - Invalid
13) Item ID < 00000, 1 <= Quantity <= 99, 0 <= Cart total <= $999.99 - Invalid
14) Item ID < 00000, 1 <= Quantity <= 99, Cart total > $999.99 - Invalid
15) Item ID < 00000, Quantity < 1, 0 <= Cart total <= $999.99 - Invalid
16) Item ID < 00000, Quantity < 1, Cart total > $999.99 - Invalid
17) Item ID < 00000, Quantity > 99, 0 <= Cart total <= $999.99 - Invalid
18) Item ID < 00000, Quantity > 99, Cart total > $999.99 - Invalid

Belongs to equivalence class -	Input Item ID	Input Quantity	Cart total	Validity Status
1	12345	34	$990.9	Valid
2	23220	12	**$1000.100**	Invalid
3	00000	**0**	$799.9	Invalid
4	99999	**-9**	**$9890**	Invalid
5	64822	**100**	$163.9	Invalid
6	55939	**103**	**$29892**	Invalid

7	**223203**	78	$49	Invalid
8	**100000**	34	**$1087.4**	Invalid
9	**555050**	**-10**	$367	Invalid
10	**230203**	**-15**	**$8912**	Invalid
11	**123929**	**150**	$738.8	Invalid
12	**99999999**	**200**	**$1689**	Invalid
13	-6	5	$678.4	Invalid
14	**-3**	12	**$1078.2**	Invalid
15	**-2**	**-1**	$549.7	Invalid
16	**-3**	**-2**	**$5625.4**	Invalid
17	**-1**	**110**	$200	Invalid
18	**-2**	**101**	**$1000.01**	Invalid

*the values in bold signify the reason that it belongs to Invalid class.

Let us now assume that a customer has a cart total of 600$ and the price for that item is 300$ and Item ID 23234.

Test Case	Input data	Expected Outcome
Quantity = 0	ID = 23234	Items with ID = 23234 will be removed from the cart. [if the item with that id was purchased previously]
Quantity = 0	ID = 23234	Error [if the item with that id was not purchased previously]