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?

Equivalence Classes:

Day:

1. D1 : Between 1 to 28
2. D2 : <1 or >31
3. D3 : 29
4. D4 : 30
5. D5 : 31

Month:

1. M1 : Months with 31 days(1, 3, 5, 7, 8,10, 12)
2. M2 : Months with 30 days(4, 6, 9, 11)
3. M3 : Month with 28/29 days (2)
4. M4 : Month < 1 or Month > 12

Year:

1. Y1 : Leap Year Between 1900 to 2015
2. Y2 : Non-Leap Year from 1900 to 2015
3. Y3 : >2015 or <1900

Test Cases identified through Equivalence Classes:

ID	DAY	MONTH	YEAR	EXPECTED OUTPUT	PROGRAM OUTPUT	TEST OUTCOME
1	D1	M1	Y1	Previous date	Previous date	Pass
2	D2	M1	Y1	Invalid	Invalid	Pass
3	D3	M1	Y1	Previous date	Previous date	Pass
4	D4	M1	Y1	Previous date	Previous date	Pass
5	D5	M1	Y1	Previous date	Previous date	Pass

6	D1	M2	Y1	Previous date	Previous date	Pass
7	D2	M2	Y1	Invalid	Invalid	Pass
8	D3	M2	Y1	Previous date	Previous date	Pass
9	D4	M2	Y1	Previous date	Previous date	Pass
10	D5	M2	Y1	Invalid	Invalid	Pass
11	D1	M3	Y1	Previous Date	Previous Date	Pass
12	D2	M3	Y1	Invalid	Invalid	Pass
13	D3	M3	Y1	Previous Date	Previous Date	Pass
14	D4	M3	Y1	Invalid	Invalid	Pass
15	D5	M3	Y1	Invalid	Invalid	Pass
16	D1	M4	Y1	Invalid	Invalid	Pass
17	D2	M4	Y1	Invalid	Invalid	Pass
18	D3	M4	Y1	Invalid	Invalid	Pass
19	D4	M4	Y1	Invalid	Invalid	Pass
20	D5	M4	Y1	Invalid	Invalid	Pass
21	D1	M1	Y2	Previous Date	Previous Date	Pass
22	D2	M1	Y2	Invalid	Invalid	Pass
23	D3	M1	Y2	Previous Date	Previous Date	Pass
24	D4	M1	Y2	Previous Date	Previous Date	Pass
25	D5	M1	Y2	Previous Date	Previous Date	Pass
26	D1	M2	Y2	Previous Date	Previous Date	Pass

27	D2	M2	Y2	Invalid	Invalid	Pass
28	D3	M2	Y2	Previous Date	Previous Date	Pass
29	D4	M2	Y2	Previous Date	Previous Date	Pass
30	D5	M2	Y2	Previous Date	Previous Date	Pass
31	D1	M3	Y2	Previous Date	Previous Date	Pass
32	D2	M3	Y2	Invalid	Invalid	Pass
33	D3	M3	Y2	Previous Date	Previous Date	Pass
34	D4	M3	Y2	Previous Date	Previous Date	Pass
35	D5	M3	Y2	Previous Date	Previous Date	Pass
36	D1	M4	Y2	Invalid	Invalid	Pass
37	D2	M4	Y2	Invalid	Invalid	Pass
38	D3	M4	Y2	Invalid	Invalid	Pass
39	D4	M4	Y2	Invalid	Invalid	Pass
40	D5	M4	Y2	Invalid	Invalid	Pass
41	D1	M1	Y3	Invalid	Invalid	Pass
42	D2	M1	Y3	Invalid	Invalid	Pass
43	D3	M1	Y3	Invalid	Invalid	Pass
44	D4	M1	Y3	Invalid	Invalid	Pass
45	D5	M1	Y3	Invalid	Invalid	Pass
46	D1	M2	Y3	Invalid	Invalid	Pass
47	D2	M2	Y3	Invalid	Invalid	Pass
48	D3	M2	Y3	Invalid	Invalid	Pass
49	D4	M2	Y3	Invalid	Invalid	Pass

50	D5	M2	Y3	Invalid	Invalid	Pass
51	D1	M3	Y3	Invalid	Invalid	Pass
52	D2	M3	Y3	Invalid	Invalid	Pass
53	D3	M3	Y3	Invalid	Invalid	Pass
54	D4	M3	Y3	Invalid	Invalid	Pass
55	D5	M3	Y3	Invalid	Invalid	Pass
56	D1	M4	Y3	Invalid	Invalid	Pass
57	D2	M4	Y3	Invalid	Invalid	Pass
58	D3	M4	Y3	Invalid	Invalid	Pass
59	D4	M4	Y3	Invalid	Invalid	Pass
60	D5	M4	Y3	Invalid	Invalid	Pass

Test Cases Identified Through Boundary Value Analysis:

ID	DAY	MONTH	YEAR	EXPECTED OUTPUT	PROGRAM OUTPUT	TEST OUTCOME
1	0	2	1904	Invalid	Invalid	Pass
2	30	2	1904	Invalid	Invalid	Pass
3	0	1	2015	Invalid	Invalid	Pass
4	32	1	2015	Invalid	Invalid	Pass
5	5	13	2000	Invalid	Invalid	Pass
6	5	5	1899	Invalid	Invalid	Pass
7	6	12	2016	Invalid	Invalid	Pass
8	3	0	2001	Invalid	Invalid	Pass
9	29	2	1904	Previous Date	Previous Date	Pass
10	29	2	1900	Invalid	Invalid	Pass
11	31	12	2015	Previous Date	Previous Date	Pass

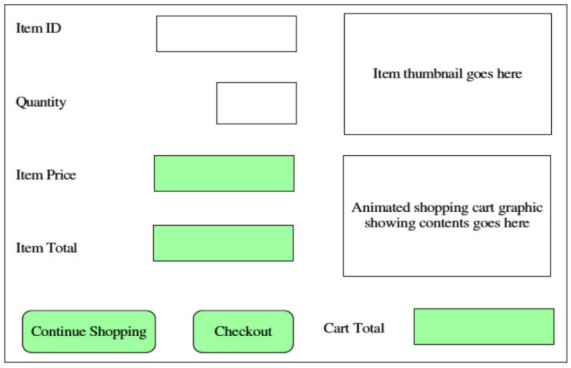
12	1	1	1900	Previous Date	Previous Date	Pass
13	31	1	1900	Previous Date	Previous Date	Pass

Code for analysis :

```
import datetime,sys

day,month,year = map(int,input().split())

leap = 1 if (((year % 4 == 0) and (year % 100 != 0)) or (year % 400 == 0)) else 0

months = {
    "1":31,"2":28,"3":31,"4":30,"5":31,
    "6":30,"7":31,"8":31,"9":30,"10":31,
    "11":30,"12":31,
}


if day<1 or str(month) not in months or year<1900 or year>2015 or day > months[str(month)]:
  if month == 2 and year>1899 and year<2016 and day==29 and leap==1:
    st = 1
  else :
    print("invalid!")
    sys.exit()

prev_month = month if day > 1 else month - 1
prev_year = year if month == prev_month else year - 1
prev_day = day - 1 if day > 1 else months[str(prev_month)]
prev_day = prev_day + leap if prev_month==2 and day==1 else prev_day
print("Previous Day : ",prev_day,datetime.datetime.strptime(str(prev_month), "%m").strftime("%B"),prev_year)
```

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 web page that accepts the orders. A screen prototype for the order-entry web page 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.

Constraints :

$00000 \leq ID \leq 99999$
$1 \leq Quantity\ (N) \leq 99$
$0 \leq Cart\ total \leq \$999.99$

Equivalence classes :

ID :

1. ID1 : between 00000-99999 (both inclusive) i.e. $00000 \leq ID \leq 99999$
2. ID2 : less that 00000 (00000 excluded) i.e. $ID < 00000$
3. ID3 : greater than 99999 (99999 excluded) i.e. $ID > 99999$

Quantity :

4. Q1 :Quantity between 0-99 (both inclusive) i.e. $0 \leq quantity \leq 99$
5. Q2 :Quantity less than 0 (0 excluded) i.e. $quantity < 0$

6. Q3 :Quantity greater than 99 (99 excluded) i.e. quantity > 99

Cart total :

7. C1 :Cart total between 0-999.99 (both inclusive) i.e. 0 ≤ cart total ≤ 999.99
8. C2 :Cart total greater than 999.99 (999.99 excluded) i.e. cart total > 999.99

Test Cases identified through Equivalence Classes:

Test case	ID	Quantity	Cart Total	Test output
1	ID1	Q1	C1	VALID
2	ID1	Q1	C2	INVALID
3	ID1	Q2	C1	INVALID
4	ID1	Q2	C2	INVALID
5	ID1	Q3	C1	INVALID
6	ID1	Q3	C2	INVALID
7	ID3	Q1	C1	INVALID
8	ID3	Q1	C2	INVALID
9	ID3	Q2	C1	INVALID
10	ID3	Q2	C2	INVALID
11	ID3	Q3	C1	INVALID
12	ID3	Q3	C2	INVALID
13	ID2	Q1	C1	INVALID
14	ID2	Q1	C2	INVALID
15	ID2	Q2	C1	INVALID
16	ID2	Q2	C2	INVALID
17	ID2	Q3	C1	INVALID
18	ID2	Q3	C2	INVALID

Boundary cases :

Test case	ID	Quantity	Cart total	Output
1	00000	10	$100	VALID
2	99999	10	$100	VALID
3	1000	0	Item removed from cart	VALID
4	1000	99	$999	VALID
5	1100	10	$999.99	VALID
6	1	10	$100	VALID
7	99998	10	$900	VALID
8	1000	1	$100	VALID
9	1000	98	$890	VALID
10	1006	86	$999.98	VALID