Q1.
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?

These classes are general(considering 31 days in each month). We are checking the validity of dates in our code(ex. months having less than 31 days).
Equivalence classes:
Day

1) day between 1 to 31 (both inclusive) valid
2) day more than 31 (31 excluded) invalid
3) day less than 1(1 not included) invalid

Month

4) month between 1 to 12 (both inclusive) valid
5) month more than 12 (12 excluded) invalid
6) month less than 1 (1 excluded) invalid

Year

7) year between 1900 to 2015 (both inclusive) valid
8) year more than 2015 (2015 excluded) invalid
9) year less than 1900 (1900 excluded) invalid

For boundary value analysis, we generated test cases like the first day of a year(1/1/year), which will change all date, month and year if we find the previous date. And similarly the first day of the month. Also, February month is an exception hence we generated those test cases as well.

Test cases and code links are given below.

Code:
https://colab.research.google.com/drive/1SWLzXUuqdbgh0S25z7jIFJo_5qH8MWu_?usp=sharing
Test Cases link:
https://drive.google.com/file/d/1fw8QLmdGIbDVPDqmdjJ1jnIB7D2IuN-K/view?usp=sharing

Q2. 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:-
→ item_ID: 00000-99999
→ Quantity: 0-99
Maximum cart total is $999.99 so,
→ Cart Total: cart_total<= $999.99

Equivalent Classes:
item_ID
1) Item_ID is between 00000-99999(Both included).
2) Item_ID is greater than 99999 (excluded).
3) Item_ID is less than 00000 (excluded).

Quantity
4) Quantity is between 0-99 (Both included).
5) Quantity is greater than 99 (excluded).
6) Quantity is less than 0 (excluded).

Cart Total
7) Cart_total is between 0-999.99$ (Both included).
8) Cart_total is greater than $999.99 (excluded).
9) Cart_total is less than 0 (excluded) which is not possible.

Code:
https://colab.research.google.com/drive/17sxbLqAhKK45ZaKhERgm5yQTh2avdCVp?usp=sharing
Test Case Link:
https://drive.google.com/file/d/1DacvOBM6xIbvf0sVkBvfYaTjxdZOWmdx/view?usp=sharing