IT 314: Software Engineering

Specification-based Test Case Generation

Group 16

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 the previous date or invalid date. Design the equivalence class test cases?

- Equivalence Partitioning:
  - Month
    1. M1: Months with exactly 31 days: Jan, Mar, May, Jul, Aug, Oct, Dec
    2. M2: Months with exactly 30 days: Apr, Jun, Sept, Nov
    3. M3: Months with 28/29 days: Feb
    4. M4: Month < 1
    5. M5: Month > 12
  - Days:
    1. D1: 1 to 28
    2. D2: 29
    3. D3: 30
    4. D4: 31
    5. D5: <1
    6. D6: >31
  - Year:
    1. Y1: Leap years from 1900 to 2015
    2. Y2: Non-leap years from 1900 to 2015
    3. Y3: >2015
    4. Y4: <1900

Possible Combinations:

ID	Day	Month	Year	Output
1	D1	M1	Y1	Previous Date
2	D2	M1	Y1	Previous Date
3	D3	M1	Y1	Previous Date
4	D4	M1	Y1	Previous Date
5	D5	ANY	ANY	Invalid
6	D6	ANY	ANY	Invalid
7	D1	M1	Y2	Previous Date
8	D2	M1	Y2	Previous Date
9	D3	M1	Y2	Previous Date
10	D4	M1	Y2	Previous Date
11	D1	M2	Y1	Previous Date
12	D2	M2	Y1	Previous Date
13	D3	M2	Y1	Previous Date
14	D4	M2	Y1	Invalid
15	ANY	ANY	Y3	Invalid
16	ANY	ANY	Y4	Invalid
17	D1	M2	Y2	Previous date
18	D2	M2	Y2	Previous date
19	D3	M2	Y2	Previous date
20	D4	M2	Y2	Invalid
21	D1	M3	Y1	Previous date
22	D2	M3	Y1	Previous date

23	D3	M3	ANY	Invalid
24	D4	M3	ANY	Invalid
25	D1	M3	Y2	Previous Date
26	D2	M3	Y2	Invalid
27	ANY	M4	ANY	Invalid
28	ANY	M5	ANY	Invalid

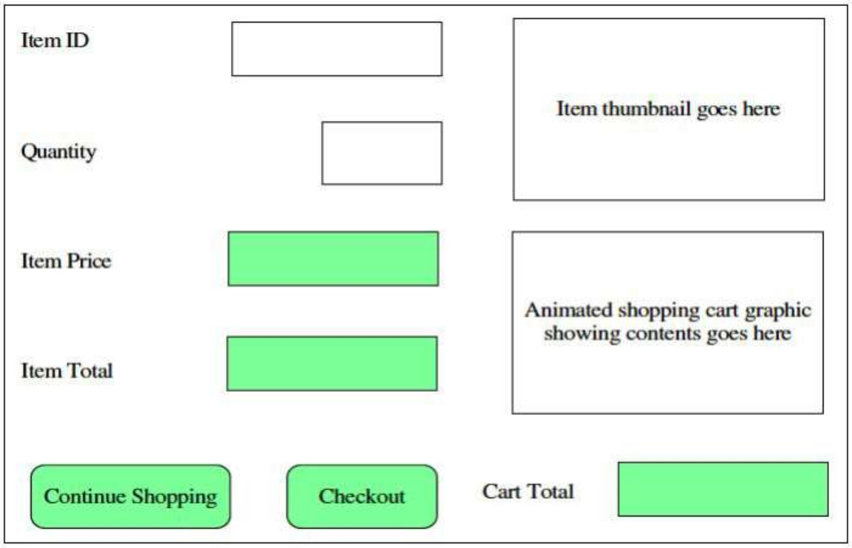
- Boundary Value Analysis:

  Test Cases:

  1) d=1, m=1, y=1900**-Valid**
  2) d=1, m=1, 1900<y<2015**-Valid**
  3) d=1, m=1, y=2015**-Valid**
  4) d=1, 1<m<12, y=1900**-Valid**
  5) d=1, 1<m<12, 1900<y<2015**-Valid**
  6) d=1, 1<m<12, y=2015**-Valid**
  7) d=1, m>12, y=1900**-Invalid**
  8) d=1, m>12, 1900<y<2015**-Invalid**
  9) d=1, m>12, y=2015**-Invalid**
  10)1<d<31, m=1, y=1900 **-Valid**
  11)1<d<31, m=1, 1900<y<2015**-Valid**
  12)1<d<31, m=1, y=2015**-Valid**
  13)1<d<31, 1<m<12, y=1900**-Valid**
  14)1<d<31, 1<m<12, 1900<y<2015**-Valid**
  15)1<d<31, 1<m<12, y=2015**-Valid**
  16)1<d<31, m>12, y=1900**-Invalid**
  17)1<d<31, m>12, 1900<y<2015 **-Invalid**
  18)1<d<31, m>12, y=2015**-Invalid**
  19)d=31, m=1, y=1900 **-Valid**
  20)d=31, m=1, 1900<y<2015**-Valid**
  21)d=31, m=1, y=2015**-Valid**
  22)d=31, 1<m<12, y=1900**-Valid**
  23)d=31, 1<m<12, 1900<y<2015**-Valid**
  24)d=31, 1<m<12, y=2015**-Valid**
  25)d=31, m>12, y=1900**-Invalid**

26)d=31, m>12, 1900<y<2015-**Invalid**
27)d=31, m>12, y=2015-**Invalid**

Q. 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
the order-entry web page is shown below.



The system accepts a five-digit numeric item Item 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 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 constraints:
Item ID: 00000-99999
Quantity: 1-99
Max cart total: less than or equal to $999.99 i.e ≤ $999.99

- Equivalence Partitioning:
  - Item ID

    1. Item ID between 00000-99999
    2. Item ID less that 00000
    3. Item ID greater than 99999
  - Quantity

    4. quantity between 0-99
    5. quantity less than 0 (0 excluded) i.e. quantity < 0
    6. quantity greater than 99 (99 excluded) i.e. quantity > 99
  - Cart total (in dollar)

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

Equivalence Class

- Valid Eq. Classes
  - 0<= ID <= 99999, Quantity = 0, Previously ordered
  - 0<= ID <= 99999, 1 <= Quantity <= 99, cart > $999.99
- Invaild Eq. Classes
  - 0<=ID<=99999, Quantity =0, previously not ordered
  - 0<=ID<=99999, 1<= Quantity <=99, cart > $999.99
  - 0<=ID<=99999, Quantity >= 99
  - ID<0, ANY
  - ID>99999, ANY
  - Quantity<0, ANY
  - Quantity>0, ANY
  - cart<0, ANY
  - cart>999.99, ANY

Let us assume that cart total is $200 (for some selected items) and the price of an item with Item Item ID 12345 is $100.

Test Case	Input Data	Expected Outcome
Item ID < 00000	-12345	Error
Item ID > 99999	100001	Error
Quantity < 0	-5	Error
Quantity > 99	101	Error
Valid ID	ID = 12345	Item Price = $100

Valid cart total	ID=12345
	quantity=7
		Cart total=$900

Invalid cart Total	ID=12345
	quantity=8
		Cart total=$1000
		(error because cart total is greater than our upper limit)

Quantity=0	ID=12345	Item with ID 12345 removed from shopping cart [if Item with ID 12345 was purchased previously]
Quantity=0	ID=12345	Error [if Item with ID 12345 was NOT purchased previously]

Valid Boundary Cases:

Item ID=99999, Quantity between 0 to 99, Cart total between 0 to 999.99
Item ID=99998, Quantity between 0 to 99, Cart total between 0 to 999.99
Item ID=0, Quantity between 0 to 99, Cart total between 0 to 999.99
Item ID=1, Quantity between 0 to 99, Cart total between 0 to 999.99
Item ID between 0 and 99999, Quantity=99, Cart total between 0 to 999.99
Item ID between 0 and 99999, Quantity=0, Cart total between 0 to 999.99
Item ID between 0 and 99999, Quantity=1, Cart total between 0 to 999.99
Item ID between 0 and 99999, Quantity between 0 to 99, Cart total=0
Item ID between 0 and 99999, Quantity between 0 to 99, Cart total=1
Item ID between 0 and 99999, Quantity between 0 to 99, Cart total=999.99
Item ID between 0 and 99999, Quantity between 0 to 99, Cart total=998.99