Capital University Of Science & Technology

Group Members

BSE181009 ALEENA AKHTAR
BSE181017 SANA MUNIR

ASSIGNMENT 3

SOFTWARE QUALITY ENGINEERING

SUBMITTED TO: Samir Obaid

Capital University Of Science & Technology

Table Of Content

Capital University Of Science & Technology

Explanation:

As industries are fast expanding, people are seeking more ways to purchase products with much ease and still maintain cost-effectiveness. Food can be ordered through the internet and payment made without going to the restaurant or the food vendor. For this system, the User will get sign up from his Number and can enter his menu ID, Quantity and password then bill will be added to their cart

Case Study:

An XYZ food company wants to develop an app that provides food delivery at your door in very little time and with the best packaging. Providing food from every famous food place near you. Order food with the best user experience. The online food ordering system is one of the latest servicers most fast-food restaurants in the western world are adopting. With this method, food is ordered online and delivered to the customer. This is made possible through the use of an electronic payment system. Customers pay with their credit cards, although credit card customers can be served even before they make payment either through cash or cheque. So, the system designed in this project will enable customers to go online and place an order for their food.

Due to the great increase in the awareness of the internet and the technologies associated with it, several opportunities are coming up on the web. So many businesses and companies now venture into their business with ease because of the internet. One such business that the internet introduced is an online food ordering system. In today's age of fast food and take out, many restaurants have chosen to focus on quick preparation and speedy delivery of orders rather than offering a rich dining experience. Until recently, most of these delivery orders were placed over the phone.

This application helps the restaurants to do all functionalities more accurately and faster way. Food Ordering System reduces manual works and improves the efficiency of restaurants. This application is helping Food Ordering s to maintain the stock and cash flows and there are many more functionalities, like.

- To store records.
- Control orders and services.
- Billings.
- Control staff and their shifting.
- Control multiple branches.
- Helps Manager to control each part of the restaurant.

For the System user will have to register to the app and will and to sign up. He has to add his credentials ***(PhoneNO, password)*** in order to register to the system, Once the user gets register to th system then they can only login by there password , system will remember there number. If a user is new he has to add his number first. Users can select Food ID from the menu and can Add it to the cart and will have to specify his food quantity.

Capital University Of Science & Technology

Functions:

1. ***Void NoOfOrder(int Quantity):***
   This function will help the user to add the number of quantities of the food that he wants to order. He has to add at least 1 item and can't add more than 10 items

2. ***Void CreateOrder(int postal_code,int Quantity,int Pin)***
   In this function, the user will enter a 5-digit postal code of his city then he will enter the quantity of the food that wants to order that should not greater than 10 and in the last, he will enter his pin again to confirm his order

3. **Void EnterPin(int Pin)**
   This function will ask the user to enter a pin then a bill will be added to their cart. User will have to enter a 3-digit pin

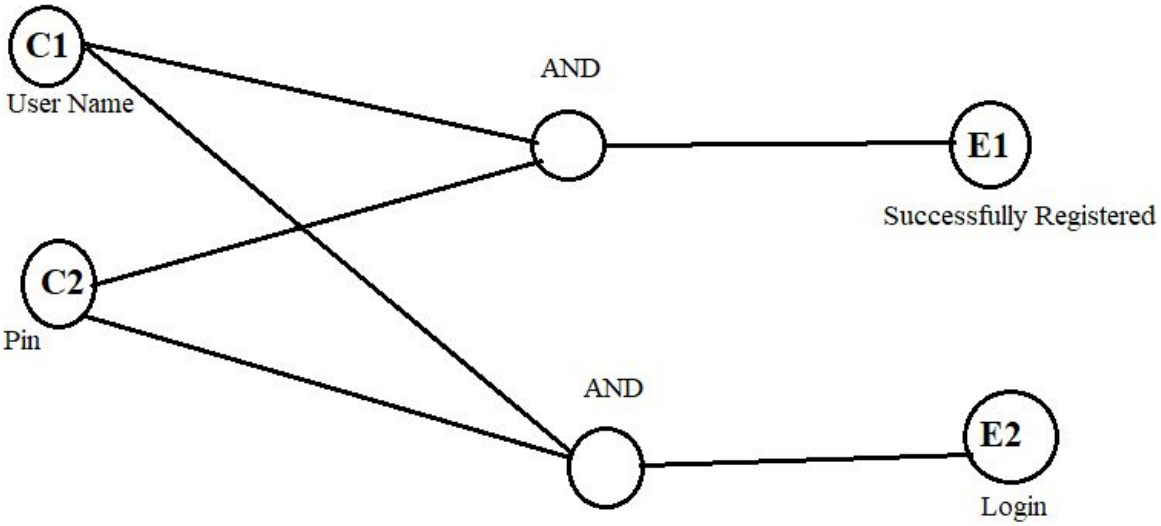
Capital University Of Science & Technology

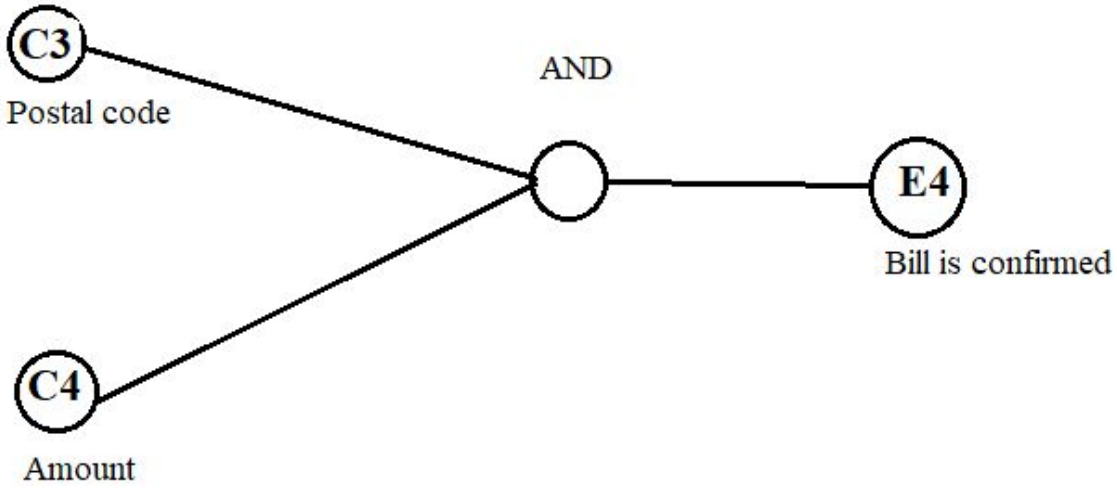
Assignment 3

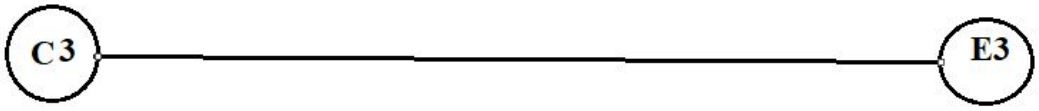
Ordering Food from System

Cause	***Effects***
C1: User Name (length >=5 && length <=8)	**E1**:Successfully Registered
C2: Correct Pin (pin >=000 && pin <=999)	**E2:** Successfully Login
C3: Correct Postal code (>=00000 to <=99999)	**E3:** Successfully postal code added
C4: Correct amount is added (amount >=1000 && <=5000)	**E4:** Bill is confirmed

Graphs


C1
User Name
C2
Pin
AND
AND
E1
Successfully Registered
E2
Login


C3
Postal code
C4
Amount
AND
E4
Bill is confirmed



Valid Postal Code

Successfully postal code added postal code added

Decision Table

Test cases	**T1**	**T2**	**T3**	**T4**
C1: User Name	1	1	0	0
C2: Pin	1	1	0	0
C3:Postal Code	0	0	1	1
C4: Amount	0	0	1	0
E1:Successfully Registered	1	0	0	0
E2: Successfully Login		1	0	0
E3: Successfully postal code added	0	0		1
E4: Bill is confirmed	0	0	1	0

Capital University Of Science & Technology

Identifying Test Cases

We are using Robust Boundary value analysis from which we get our expected results

Test cases	**Input(cause)**		**Expected Output (effects)**
	User name	**Pin**	
T1	abcdefg	123	**E1**:Successfully Registered
T2	adcfr	234	**E2**: Successfully Login

	Postal Code	
T3	56466	**E3**: Successfully postal code added

	Postal Code	**Amount**	
T4	54666	2300	**E4:** Bill is confirmed

Capital University Of Science & Technology

Assignment no 2

Mistake of Assignment 2	According to feedback removing OrderID and specifying limit for pin

Equivalence Class Partitioning

Strong Robust Equivalence Class:

In Equivalence Class Partitioning we will see working of Strong Robust Equivalence Class. In this function we test beyond the boundaries. Strong robust testing produces test cases for all valid and invalid elements of the product of the equivalence class. However, it is incapable of reducing the redundancy in testing.

1. ***Void NoOfOrder(int Quantity)***
   For Quantity
   - Below boundary (less 1)
   - Within boundary (1-10)
   - Beyond boundary (greater than 10)

Case	Quantity	Expected output
1	0	Invalid
2	1	Valid
3	11	Invalid

2. **Void Enterpin(int Pin)**
   As pin is integer user can add only digits, this system will take only 3-digit pin only
   - Below boundary (less than 000)
   - Within boundary (000-999)
   - Beyond boundary (greater than 999)

Case	Pin	Expected output
1	12	Invalid
2	121	Valid
3	1000	Invalid

3. *Void CreateOrder(int postal_code ,int Quantity,int Password)*

As pin is integer user can add only digits, this system will take only 3-digit pin only

- Below boundary (less than 000)
- Within boundary (000-999)
- Beyond boundary (greater than 999)

For Quantity

- Below boundary (less 1)
- Within boundary (1-10)
- Beyond boundary (greater than 10)

As Postal code of city is integer, we will take 5-digit input only

- Below boundary (less 00000) i.e 1234
- Within boundary (from 00000-99999)
- Beyond boundary (greater than 99999) i.e 100000

No of Case	Test cases	Expected output
1	1234,0,123	Invalid
2	1234,10,123	invalid
3	1234,0,0000	Invalid
4	1234,10,0000	Invalid
5	00000,10,0000	Valid
6	00000,0,123	Invalid
7	00000,10,123	Invalid

8	00000,0,0000	valid
9	100000,13,10000	Invalid
10	100000,1,10000	Invalid
11	100000,13,10000	Invalid
12	00000,13,10000	Invalid
13	00000,13,0000	Invalid
14	100000,1,0000	Invalid
15	00000,1,10000	Invalid

Capital University Of Science & Technology

Assignment no 1

Mistake in Assignment 1	
1	In **BVA** we added random numbers, but have add within Boundaries
2	In **RBVA** we added completely wrong and random test cases
3	In **worst case BVA** we added completely wrong and random test cases
4	In **Robust worst case BVA** we added completely wrong and random test cases

BLACK BOX TESTING

BVA:

In **BVA** testing we test every single possible combination. Cases are calculated by the formula 4(n)+1 where n is the number of variables.

1. **Void Enterpin(int Pin)**
   Total test cases 4(1)+1=5,

   As pin is integer user can add only digits, this system will take only 3-digit pin only
   - min = 000
   - min+1 =001
   - normal =546
   - max-1 = 998,
   - max =999

Case	Pin	Expected output
1	000	Valid
2	001	Valid

3	546	Valid
4	998	Valid
5	999	Valid

2. ***Void NoOfOrder(int Quantity)***
   Total test cases 4(1)+1=5,

   For Quantity
   - min = 1
   - min+1 =2
   - normal =5
   - max-1 = 9
   - max =10

Case	**Quantity**	**Expected output**
1	1	Valid
2	2	Valid
3	5	Valid
4	9	Valid
5	10	Valid

3. ***Void CreateOrder(int Postal_code,int Quantity,int Password)***
   Total test cases 4(3)+1=13,

   For Quantity
   - min = 1
   - min+1 =2
   - normal =5
   - max-1 = 9

- max =10

As pin is integer user can add only digits, this system will take only 3-digit pin only

- min = 000
- min+1 =001
- normal =546
- max-1 = 998,
- max =999

As Postal code of city is integer, we will take 5-digit input only

- min = 00000
- min+1 =00001
- normal =54666
- max-1 = 99998,
- max =99999

No Of Cases	Test cases	Expected Output
1	5,000,00000	Valid
2	5,001,00001	Valid
3	5,546,54666	Valid
4	5,998,99998	Valid
5	5,999,99999	Valid
6	001,1,00000	Valid
7	001,2,00001	Valid
8	001,5,54666	Valid
9	001,9,99998	Valid
10	001,10,99999	Valid
11	54666,2,001	Valid
12	54666,9,998	Valid
13	54666,10,999	Valid

Capital University Of Science & Technology

RBVA:

In **RBVA** testing we test every single possible combination. Cases are calculated by the formula 6(n)+1 where n is the number of variables.

1. **Void Enterpin(int Pin)**
   Total test cases 6(1)+1=7,
   As pin is integer user can add only digits, this system will take only 3-digit pin only
   - min-1=99
   - min = 000
   - min+1 =001
   - normal =546
   - max-1 = 998,
   - max =999
   - max+1=1000

Case	Pin	Expected output
1 (below)	99	Invalid
2	000	valid
3	001	valid
4	546	valid
5	998	valid
6	999	valid
7 (above)	1000	invalid

2. ***Void NoOfOrder(int Quantity)***
   Total test cases 6(1)+1=7, only binary values will be valid

   For Quantity

- min -1 =0
- min = 1
- min+1 =2
- normal =5
- max-1 = 9
- max =10
- max+1=11

Case	Quantity	Expected output
1 (below)	0	Invalid
2	1	valid
3	2	valid
4	5	valid
5	9	valid
6	10	valid
7 (above)	11	invalid

3. ***Void CreateOrder(int Postal_code,int Quantity,int Password)***

Total test cases 6(3)+1=19,

For Quantity

- min -1 =0
- min = 1
- min+1 =2
- normal =5
- max-1 = 9
- max =10
- max+1=11

As pin is integer user can add only digits, this system will take only 3-digit pin only

- min-1=99
- min = 000
- min+1 =001
- normal =546
- max-1 = 998,
- max =999
- max+1=1000

As Postal code of city is integer, we will take 5-digit input only

- min-1=9999
- min = 00000
- min+1 =00001
- normal =54666
- max-1 = 99998,
- max =99999
- max+1=100000

No of Case	**Test cases**	**Expected output**
1	0,99,9999	invalid
2	0,99,00000	invalid
3	0,99,00001	invalid
4	0,99,54666	invalid
5	0,99,99998	invalid
6	0,99,99999	invalid
7	0,99,100000	invalid
8	0,000,9999	invalid
9	0,000,00000	invalid
10	0,000,00001	invalid
11	0,000,54666	invalid
12	0,000,99998	invalid
13	0,000,99999	invalid
14	0,000,100000	invalid

15	1,001,9999	invalid
16	1,001,00000	valid
17	1,001,00001	valid
18	1,001,54666	Valid
19	1,001,99998	valid

Worst-Case BVA:

In **Worst-Case BVA** testing we test every single possible combination. Cases are calculated by the formula 5^n (5 power n) where n is the number of variables. To generate test cases first we choose 5 numbers between the given boundary values (min, min+1, normal, max-1, max).

1. **Void Enterpin(int Pin)**
   Total test cases 5^1=5,

   As pin is integer user can add only digits, this system will take only 3-digit pin only
   - min = 000
   - min+1 =001
   - normal =546
   - max-1 = 998,
   - max =999

Case	Pin	Expected output
1	000	Valid
2	001	Valid
3	546	Valid
4	998	Valid
5	999	Valid

2. ***Void NoOfOrder(int Quantity)***

Total test cases 5^1=5

For Quantity

- min = 1
- min+1 =2
- normal =5
- max-1 = 9
- max =10

Case	Quantity	Expected output
1	1	Valid
2	2	Valid
3	5	Valid
4	9	Valid
5	10	Valid

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***

Total test cases 5^3=125,

As pin is integer user can add only digits, this system will take only 3-digit pin only

- min = 000
- min+1 =001
- normal =546
- max-1 = 998,
- max =999

For Quantity

- min = 1
- min+1 =2
- normal =5
- max-1 = 9
- max =10

As Postal code of city is integer, we will take 5-digit input only

- min = 00000

- min+1 =00001
- normal =54666
- max-1 = 99998,
- max =99999

No of cases	Test cases	Expected Output
1	1,000,00000	Valid
2	1,001,00001	Valid
3	1,546,54666	Valid
4	1,998,99998	Valid
5	1,999,99999	Valid
6	2,000,00000	Valid
7	2,001,00001	Valid
8	2,546,54666	Valid
9	2,998,99998	Valid
10	2,999,99999	Valid
11	5,000,00000	Valid
12	5,001,00001	Valid
13	5,546,54666	Valid
14	5,998,99998	Valid
15	5,999,99999	Valid
16	9,000,00000	Valid
17	9,001,00001	Valid
18	9,546,54666	Valid
19	9,998,99998	Valid
20	9,999,99999	Valid
21	10,000,00000	Valid

22	10,001,00001	Valid
23	10,546,54666	Valid
24	10,998,99998	Valid
25	10,999,99999	Valid
26	000,2,00001	Valid
27	000,5,54666	Valid
28	000,9,99998	Valid
29	000,10,99999	Valid
30	001,1,00000	Valid
31	001,5,54666	Valid
32	001,9,99998	Valid
33	001,10,99999	Valid
34	546,1,00000	Valid
35	546,2,00001	Valid
36	546,9,99998	Valid
37	546,10,99999	Valid
38	998,1,00000	Valid
39	998,2,00001	Valid
40	998,5,54666	Valid
41	998,10,99999	Valid

42	999,1,00000	Valid
43	999,2,00001	Valid
44	999,5,54666	Valid
45	999,9,99998	Valid
46	000,2,00001	Valid
47	000,5,54666	Valid

48	000,9,99998	Valid
49	000,10,99999	Valid
50	001,1,00000	Valid
51	001,5,54666	Valid
52	001,9,99998	Valid
53	001,10,99999	Valid
54	546,1,00000	Valid
55	546,2,00001	Valid
56	546,9,99998	Valid
57	546,10,99999	Valid
58	998,1,00000	Valid
59	998,2,00001	Valid
60	998,5,54666	Valid
61	998,10,99999	Valid
62	999,1,00000	Valid

Robust Worst-Case BVA:

In **Robust Worst-Case BVA** testing we test every single possible combination. Cases are calculated by the formula 7^n (7 power n) where n is the number of variables. To generate test cases first we choose 5 numbers between the given boundary values (min-1, min, min+1, normal, max-1, max, max+1).

1. **Void Enterpin(int Pin)**

   Total test cases 7^1=7

   As pin is integer user can add only digits, this system will take only 3-digit pin only
   - min-1=99
   - min = 000
   - min+1 =001
   - normal =546
   - max-1 = 998,

- max =999
- max+1=1000

Case	Pin	Expected output
1	99	Invalid
2	000	Valid
3	001	Valid
4	546	Valid
5	998	Valid
6	999	Valid
7	1000	Invalid

2. ***Void NoOfOrder(int Quantity)***
   Total test cases 7^1=7,

   For Quantity
   - min -1 =0
   - min = 1
   - min+1 =2
   - normal =5
   - max-1 = 9
   - max =10
   - max+1=11

Case	Quantity	Expected output
1	0	Invalid

2	1	Valid
3	2	Valid
4	5	Valid
5	9	Valid
6	10	Valid
7	11	Invalid

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***

Total test cases 7^3=343,

As pin is integer user can add only digits, this system will take only 3-digit pin only

- min-1=99
- min = 000
- min+1 =001
- normal =546
- max-1 = 998,
- max =999
- max+1=1000

For Quantity

- min -1 =0
- min = 1
- min+1 =2
- normal =5
- max-1 = 9
- max =10
- max+1=11

As Postal code of city is integer, we will take 5-digit input only

- min-1=9999
- min = 00000
- min+1 =00001
- normal =54666
- max-1 = 99998,
- max =99999

- max+1=100000

No of cases	**Test cases**	**Expected output**
1	0,99,9999	Invalid
2	0,99,00000	Invalid
3	0,99,00001	Invalid
4	0,99,54666	Invalid
5	0,99,99998	Invalid
6	0,99,99999	Invalid
7	0,99,100000	Invalid
8	0,000,9999	Invalid
9	0,000,00000	valid
10	0,000,00001	valid
11	0,000,54666	valid
12	0,000,99998	valid
13	0,000,99999	valid
14	0,000,100000	Invalid
15	0,001,9999	Invalid
16	0,001,00000	valid
17	0,001,00001	valid
18	0,001,54666	valid
19	0,001,99998	valid
20	0,001,99999	valid
21	0,001,100000	Invalid
22	0,546,9999	Invalid
23	0,546,00000	valid
24	0,546,00001	valid

25	0,546,54666	valid
26	0,546,99998	valid
27	0,546,99999	valid
28	0,546,100000	Invalid
29	0,998,99999	Invalid
30	0,998,00000	valid
31	0,998,00001	valid
32	0,998,54666	valid

33	0,998,99998	valid
34	0,998,99999	valid
35	0,998,100000	Invalid
36	0,99,9999	Invalid
37	0,99,00000	Invalid
38	0,99,00001	Invalid
39	0,99,54666	Invalid
40	0,99,99998	Invalid
41	0,99,99999	Invalid
42	0,99,100000	Invalid
43	0,1000,9999	Invalid
44	0,1000,00000	Invalid
45	0,1000,00001	Invalid
46	0,1000,54666	Invalid
47	0,1000,99998	Invalid
48	0,1000,99999	Invalid
49	0,1000,100000	Invalid
50	1,99,9999	Invalid

51	1,99,00000	Invalid
52	1,99,00001	Invalid
53	1,99,54666	Invalid
54	1,99,99998	Invalid
55	1,99,99999	Invalid
56	1,99,100000	Invalid
57	1,000,9999	Invalid
58	1,000,00000	valid
59	1,000,00001	valid
60	1,000,54666	valid
61	1,000,99998	valid
62	1,000,99999	valid
63	1,000,100000	Invalid
67	1,001,9999	Invalid
68	1,001,00000	valid
69	1,001,00001	valid
70	1,001,54666	valid
71	1,001,99998	valid
72	1,001,99999	valid
73	1,001,100000	Invalid
74	1,546,9999	Invalid
76	1,546,00000	valid
77	1,546,00001	valid
78	1,546,54666	valid
79	1,546,99998	valid
80	1,5496,99999	valid
81	1,546,100000	Invalid

82	1,998,9999	Invalid
83	1,998,00000	valid

84	1,998,00001	valid
85	1,998,54666	valid
86	1,998,99998	valid
87	1,998,99999	valid
88	1,998,100000	Invalid
89	1,999,9999	Invalid
90	1,999,00000	valid
91	1,999,00001	valid
92	1,999,54666	valid
93	1,999,99998	valid
94	1,999,99999	valid
95	1,999,100000	Invalid
96	1,1000,9999	Invalid
97	1,1000,00000	Invalid
98	1,1000,00001	Invalid
99	1,1000,54666	Invalid
100	1,1000,99989	Invalid
101	1,1000,99999	Invalid
102	1,1000,100000	Invalid
103	2,99,9999	Invalid
104	2,99,00000	Invalid
105	2,99,00001	Invalid
106	2,99,54666	Invalid
107	2,99,99998	Invalid

108	2,99,99999	Invalid
109	2,99,100000	Invalid
110	2,000,9999	Invalid
111	2,000,00000	valid
112	2,000,00001	valid
113	2,000,54666	valid
114	2,000,99998	valid
115	2,000,99999	Invalid
116	2,000,100000	Invalid
117	2,001,9999	valid
118	2,001,00000	valid
119	2,001,00001	valid
120	2,001,54666	valid
121	2,001,99998	valid
122	2,001,99999	valid
123	2,001,100000	Invalid
124	2,546,9999	Invalid
125	2,546,00000	valid
126	2,546,00001	valid
127	2,546,54666	valid
128	2,546,99998	valid
129	2,546,99999	valid
130	2,546,100000	Invalid
131	2,998,9999	Invalid
132	2,998,00000	valid
133	2,998,00001	valid

134	2,998,54666	valid
135	2,998,99998	valid
136	2,998,99999	Invalid
137	2,998,100000	Invalid
138	2,999,9999	valid
139	2,999,00000	valid
140	2,999,00001	valid
141	2,999,54666	valid
142	2,999,99998	valid
143	2,999,99999	Invalid
144	2,999,100000	Invalid
145	2,1000,9999	Invalid
146	2,1000,00000	Invalid
147	2,1000,00001	Invalid
148	2,1000,54666	Invalid
140	2,1000,99998	Invalid
150	2,1000,99999	Invalid
151	2,1000,100000	Invalid
152	5,99,9999	Invalid

153	5,99,00000	Invalid
154	5,99,00001	Invalid
156	5,99,54666	Invalid
157	5,99,99998	Invalid
158	5,99,99999	Invalid
159	5,99,100000	Invalid

160	5,000,9999	Invalid
161	5,000,00000	valid
162	5,000,00001	valid
163	5,000,54666	valid
164	5,000,99998	valid
165	5,000,99999	valid
166	5,0000,10000	valid
167	5,001,9999	valid
168	5,001,00000	valid
169	5,001,00001	valid
170	5,001,54666	valid
171	5,001,99998	valid
172	5,001,99999	valid