A_2.1.7_Classes_and_Boundaries

1953067 宋潇歌　25%

2052225 张勤杭　25%

2051840 梁厚　25%

1950389 季艺　25%

Question

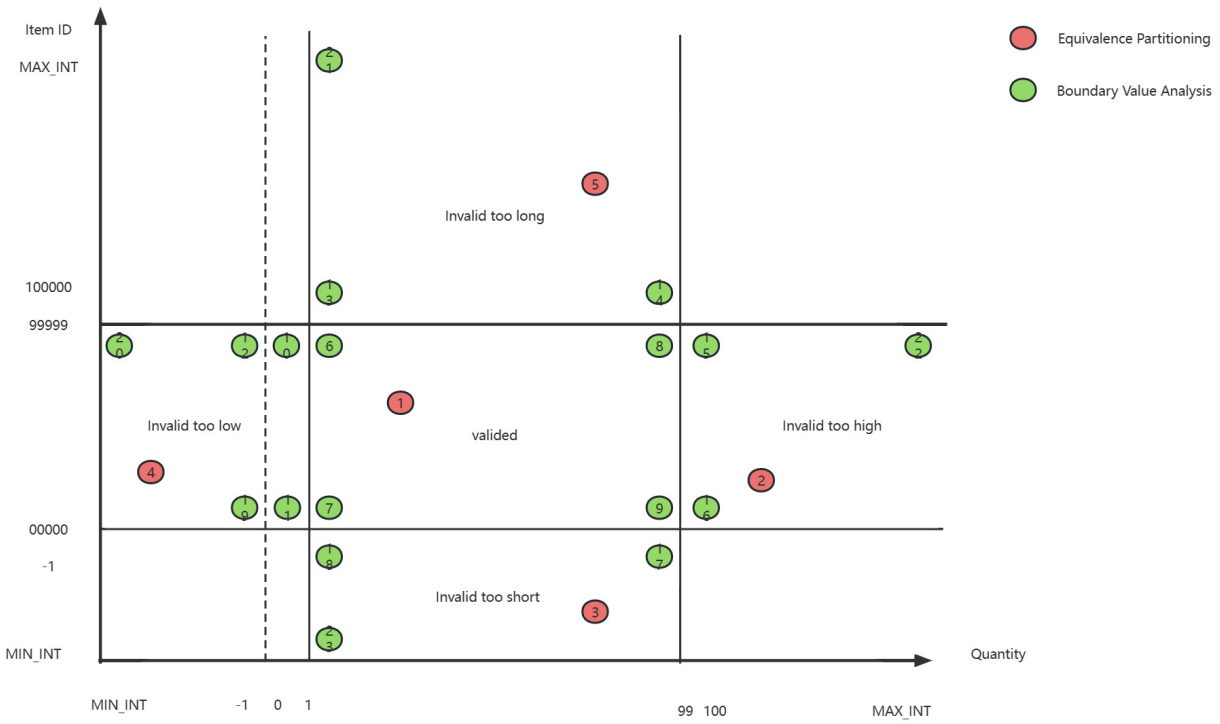
- You are testing an e-commerce site that sells Omninet knick-knacks like baseball caps, jackets, etc.
- Create functional tests for accepting orders
  - The system accepts a five-digit numeric item ID number from 00000 to 99999 (integer / string?)
  - Item IDs are sorted by price, with the cheapest items having the lower (close to 00000) item ID numbers and the most expensive items having the higher (close to 99999) item ID numbers
  - 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
  - The maximum total order is $999.99
- Use boundary value analysis and equivalence class partitioning to create tests in the following template
- Discuss

Answer

Overview

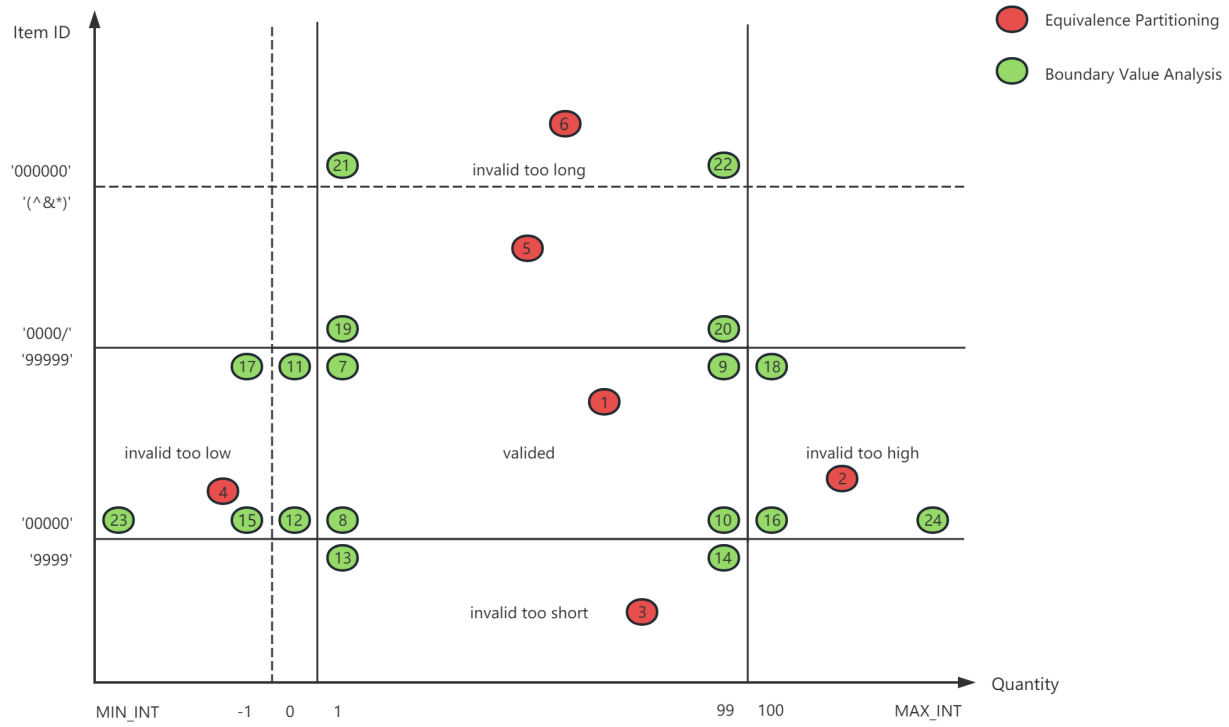
According to the question, there are two inputs (item ID and quantity), and the data type of item ID is unknown, so we offer two testing solutions (when item ID is string and when item ID is integer).

When item ID is integer, we have:



- In Equivalence Partitioning, we plan to conduct 5 functional tests (one testing matches one area in the aforementioned chart). However, we do not test the cases in the white area, because we think that the frequency of occurrence is too low.
- In Boundary Value Analysis, we plan to conduct 4 tests to validate the boundaries of the Valided area. And 2 tests for validating the cases where quantity is equal to 0 (one is validated, because previously-ordered item ID existed. Another is not). Then 8 testing cases in the 'Invalided too long', 'Invalided too short', 'Invalid too high' and 'Invalid too low' area. Finally, we have to test the cases where input (item ID and quantity) beyond MAX_INT and MIN_INT.

When item ID is string, we have:



- In Equivalence Partitioning, we plan to conduct 6 functional tests (one testing matches one area in the aforementioned chart).
- In Boundary Value Analysis, since string does not have MAX and MIN, we do not need to validate MAX and MIN of item ID. Instead, when the length of string is five and contains non-numeric characters, the input is also invalid, so we test two more cases in the 'invalid too long' area. Other testing cases are similar to the testing solution when item ID is integer.

Total Order

Since the price of a specific item is unknown, and the total order could not be larger than $999.99, we have to conduct another testing solution like:

- In Equivalence Partitioning, we plan to conduct 3 functional tests (one testing matches one area in the aforementioned chart).
- In Boundary Value Analysis, we have 6 testing cases, including 0, 999.99, -1, 1000, MIN_INT, MAX_INT.

Full Cart

Finally, it is necessary to consider the case where the cart is full. So another testing case will be needed.

Testing Case

Integer

i. Equivalence Partitioning, testing 5 points in 5 different classes.

Test Number	1.1	1.2	1.3	1.4	1.5
Inputs, Action					
Item ID	70000	00500	-50	00100	666666
Quantity	20	110	70	-60	80
Con. Shopping	Y	Y	Y	Y	N
Check out	N	N	N	N	Y
Expected Result					
Item Price					
Cart Contents	70000*20	-	-	-	70000*20
Cart Total	70000Price* 20	-	-	-	70000Price* 20
Wrong: quantity invalidated		√		√	
Wrong: item ID invalidated			√		√
Wrong: total price too large					
Wrong: quantity of this item could not be 0					

ii. 4 points in the 4 corners of the 'valided' area and one legal 0 case.

Test Number	1.6	1.7	1.8	1.9	1.10
Inputs, Action					
Item ID	99999	00000	99999	00000	99999

Quantity	1	1	99	99	0
Con. Shopping	Y	Y	Y	Y	N
Check out	N	N	N	N	Y
Expected Result					
Item Price					
Cart Contents	99999*1	00000*1	99999*99	00000*99	-
Cart Total	99999Price*1	00000Price*1+99999Price*1	00000Price*1+99999Price*1+99999Price*99	00000Price*1+99999Price*1+99999Price*99+00000Price*99	00000Price*1+00000Price*99
Wrong: quantity invalidated					
Wrong: item ID invalidated					
Wrong: total price too large					
Wrong: quantity of this item could not be 0					

iii. Illegal 0 and 8 testing cases in separated corners.

Test Number	1.11	1.12	1.13	1.14	1.15
Inputs, Action					
Item ID	00000	99999	100000	100000	99999
Quantity	0	-1	1	99	100
Con. Shopping	Y	Y	Y	Y	Y
Check out	N	N	N	N	N
Expected Result					
Item Price					

Cart Contents	-	-	-	-	-
Cart Total	-	-	-	-	-
Wrong: quantity invalidated		√			√
Wrong: item ID invalidated			√	√	
Wrong: total price too large					
Wrong: quantity of this item could not be 0	√				

Test Number	1.16	1.17	1.18	1.19	
Inputs, Action					
Item ID	00000	-1	-1	00000	
Quantity	100	99	1	-1	
Con. Shopping	Y	Y	Y	N	
Check out	N	N	N	Y	
Expected Result					
Item Price					
Cart Contents	-	-	-	-	
Cart Total	-	-	-	-	
Wrong: quantity invalidated	√			√	
Wrong: item ID invalidated		√	√		
Wrong: total price too large					

Wrong: quantity of this item could not be 0					

iv. 4 cases to validate the boundary of data type (INT)

Test Number	1.20	1.21	1.22	1.23	
Inputs, Action					
Item ID	99999	MAX_INT	99999	MIN_INT	
Quantity	MIN_INT	1	MAX_INT	1	
Con. Shopping	Y	Y	Y	N	
Check out	N	N	N	Y	
Expected Result					
Item Price					
Cart Contents	-	-	-	-	
Cart Total	-	-	-	-	
Wrong: quantity invalidated	√		√		
Wrong: item ID invalidated		√		√	
Wrong: total price too large					
Wrong: quantity of this item could not be 0					

String

i. Equivalence Partitioning, testing 6 points in 6 different classes.

Test Number	2.1	2.2	2.3	2.4	2.5	2.6
Inputs, Action						
Item ID	70000	03000	9	02000	3#5%)	1412&3
Quantity	80	150	70	-60	50	60
Con. Shopping	Y	Y	Y	Y	Y	N
Check out	N	N	N	N	N	Y
Expected Result						
Item Price						
Cart Contents	70000*20	-	-	-	-	70000*20
Cart Total	70000 Price*20	-	-	-	-	70000Price*20
Wrong: quantity invalidated		√		√		
Wrong: item ID invalidated			√		√	√
Wrong: total price too large						
Wrong: quantity of this item could not be 0						

ii. 4 points in the 4 corners of the 'valided' area and one legal 0 case.

Test Number	2.7	2.8	2.9	2.10	2.11
Inputs, Action					
Item ID	99999	00000	99999	00000	99999
Quantity	1	1	99	99	0
Con. Shopping	Y	Y	Y	Y	N

Check out	N	N	N	N	Y
Expected Result					
Item Price					
Cart Contents	99999*1	00000*1	99999*99	00000*99	-
Cart Total	99999Price*1	00000Price*1+99999Price*1	00000Price*1+99999Price*1+99999Price*99	00000Price*1+99999Price*1+99999Price*99+00000Price*99	00000Price*1+00000Price*99
Wrong: quantity invalidated					
Wrong: item ID invalidated					
Wrong: total price too large					
Wrong: quantity of this item could not be 0					

iii. Illegal 0 and 10 testing cases in separated corners.

Test Number	2.12	2.13	2.14	2.15	2.16	2.17
Inputs, Action						
Item ID	00000	9999	9999	00000	00000	99999
Quantity	0	1	99	-1	100	-1
Con. Shopping	Y	Y	Y	Y	Y	Y
Check out	N	N	N	N	N	N
Expected Result						
Item Price						
Cart Contents	-	-	-	-	-	-
Cart Total	-	-	-	-	-	-

Wrong: quantity invalidated				√	√	√
Wrong: item ID invalidated		√	√			
Wrong: total price too large						
Wrong: quantity of this item could not be 0	√					

Test Number	2.18	2.19	2.20	2.21	2.22
Inputs, Action					
Item ID	99999	0000/	0000/	000000	000000
Quantity	100	1	99	1	99
Con. Shopping	Y	Y	Y	Y	N
Check out	N	N	N	N	Y
Expected Result					
Item Price					
Cart Contents	-	-	-	-	-
Cart Total	-	-	-	-	-
Wrong: quantity invalidated	√				
Wrong: item ID invalidated		√	√	√	√
Wrong: total price too large					
Wrong: quantity of this item					

could not be 0					

iv. 2 cases to validate the boundary of data type (INT)

Test Number	2.23	2.24			
Inputs, Action					
Item ID	00000	00000			
Quantity	MIN_INT	MAX_INT			
Con. Shopping	Y	N			
Check out	N	Y			
Expected Result					
Item Price					
Cart Contents	-	-			
Cart Total	-	-			
Wrong: quantity invalidated	√	√			
Wrong: item ID invalidated					
Wrong: total price too large					
Wrong: quantity of this item could not be 0					

Total Order Testing

Since the price of a specific item is unknown, and the total order could not be larger than $999.99, we have to validate:

1. Total order = 0;
2. Total order = 999.99;
3. Total order = -1;

4. Total order = 1000;
5. Total order = MIN_INT;
6. Total order = MAX_INT.

Full Cart Testing

Test the case where the cart is full.