Helloworld Bookstore

1.0 Specification

The Hello World bookstore has set out to change their sales strategies to attract more customers before Halloween.

1.1) On the eve of the campaign, the bookstore has books in 3 categories: books in a general category, second-hand books and Halloween-themed books.

1.2) Books in a general category are € 10, second-hand books are € 5 and Halloween-themed books are € 7.5. These books have fixed prices.

1.3) Generally, customers get a 10% discount if they make a purchase of € 25 or more, a 15% discount if they make a purchase of € 45 or more, and a 20% discount if they make a purchase of € 75 or more. However, customers must purchase at least 1 Halloween-themed book to receive this discount.

1.4) If a customer buys 3 or more books in a general category, 1 Halloween book is free. (This gift book is not considered for a general discount)

1.5) If a customer buys 6 or more second-hand books, 1 Halloween book is free. (This gift book is not considered for a general discount)

1.6) 1.4 and 1.5 are invalid at the same time. However, one of them must be applied.

1.7) According to the general rules, each VIP customer receives a 2% discount on each book when they purchase books in a general category.

1.8) The percentage of the total discount cannot exceed 25%.

2.0 Test design documentation

JavaScript is called a dynamic language because it does not just have a few dynamic aspects, pretty much everything is dynamic. All variables are dynamic (both in type and existence), and even the code is dynamic. Therefore, the types of parameters entered into the **constructor** must be tested first.

2.1 constructor(books, isUserVIP)

Constructor requires two parameters (`books` and `isUserVIP`). Type of the `books` parameter is array and each array object requires `price` and `category` keys whilst `price` is non-negative number and `category` is string. Type of the `isUserVIP` parameter is boolean.

category	price	isUserVIP	Expected result
undefined	undefined	false	Books DB should be include array where length > 0
undefined	10	false	Books DB should be include category
null	undefined	false	Books DB should be include non-empty price
null	null	false	Books DB should be defined
false	1	false	Category should be string
General	undefined	false	Books DB should be include price
undefined	null	false	Books DB should be include non-empty category
General	PriceString	false	Price should be number
General	-10	false	Price should be a non-negative number
undefined	undefined	undefined	Books and price should be included
General	0	undefined	isUserVIP should be included
General	10	null	isUserVIP should be non-empty
General	0	boolean	isUserVIP should be boolean not string
General	7.5	false	Success
General	7.5	true	Success

2.2 addToCart(bookID)

First of all, booksDB should be defined. As can be seen, id of the General book is 0, Second Hand book is 1, Halloween book is 2.

```
const booksDB = [
    { category: "General", price: 10 },
    { category: "Second Hand", price: 5 },
    { category: "Halloween", price: 7.5 },
];
```

Next, items should define a books on cart. Thus, in this step, total price of books in cart and free halloween book should be verified (1.4, 1.5).

items	Free halloween book	Expected result
[0, 0, 0]	1	1
[1, 1, 1, 1, 1, 1]	1	1
[0, 0, 0, 1, 1, 1, 1, 1, 1]	1	1
[0]	0	0

Total price should be verified. Note: in this case, free halloween book should not considered as a part of campaign, because, the price after discount should be verifed in another test case.

items	Total price	Expected result
[0, 1, 1, 1, 0, 2, 0]	52.5	Total price should be equally
[]	0	Total price should be 0

As already mentioned, it is necessary to check the arguments of each function.

Book id	**Expected result**
100	Book id should be in range
undefined	Book id should be non-empty
null	Book id should be not null
-1	Book id should be less than 0
Hello World	Book id should be integer

2.3 classifyBooksOnCart()

In this step, the number of the books on the cart should be verified when a discount is applied. Red coloured asterisk (*) determines the test when a discount is applied.

items	**general**	**secondHand**	**halloween**
[0, 1, 2]	1	1	1
[]	0	0	0
[0, 0, 0]	3	0	0
[0, 0, 0] *	3	0	1
[1, 1, 1, 1, 1, 1]	0	6	0
[1, 1, 1, 1, 1, 1] *	0	6	1

2.4 applyDiscountGetPercentage()

In this step, the main purpose of the test is checking a percentage of discount (1.3). This function return a percentage.

items	**isUserVIP**	**percentage**
[]	true	0
[0]	true	2
[0, 0, 0]	true	6
[0, 0, 0, 2]	true	16
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ..., 0]	true	25
[]	false	0
[0, 0, 0]	false	0
[0, 0, 0, 2]	false	10
[0, 0, 2, 0]	false	15
[0, 0, 2, 0, 0, 0, 0]	false	20
[1, 1, 1, 1, 1, 1, 2]	false	10
[1, 1, 1, 1, 1, 1, 1, 1, 2]	false	15
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2]	false	20

Items on the table aimed to reach the boundaries of the percentage limit.

2.5 getDiscountedPrice()

In this section, It is possible to see how the test found the bug. In 2.4 section, the test shows the correct percentage results, however, in this case, it is shown that the test is able to finds bugs easily. This function return a discounted total price.

items	isUserVIP	Discounted price
[]	true	0
[0]	true	total - ((total*2)/100)
[0, 0, 0]	true	total - ((total*6)/100)
[0, 0, 0, 2]	true	total - ((total*16)/100)
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ..., 0]	true	total - ((total*25)/100)
[]	false	0
[0, 0]	false	total
[1, 1, 1, 1, 1]	false	total
[0, 0, 0]	false	total
[1, 1, 1, 1, 1]	false	total
[0, 0, 0, 2]	false	total - ((total*10)/100)
[0, 0, 0, 2, 0]	false	total - ((total*15)/100)
[0, 0, 0, 2, 0, 0, 0, 0]	false	total - ((total*20)/100)
[1, 1, 1, 1, 1, 1, 2]	false	total - ((total*10)/100)
[1, 1, 1, 1, 1, 1, 1, 1, 2]	false	total - ((total*15)/100)
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2]	false	total - ((total*20)/100)

3.0 Bugs

It was necessary to enter 2 bugs into the code. The pre-defined bugs are given as follows.

3.1 Type of the price

The price of each book should not be less than zero. If the value of any book is less than zero, an error must be made inside the `contructor`.

3.2 Price of the free book

When one general category or six second hand category books are added to the cart, 1 free Halloween book must be added. However, the price of the book remains the same and affects the overall percentage.

4.0 Coverage report

It is difficult to achieve 100% code coverage for a large-scale project. As this project is not large-scale, one of the goals was to achieve 100% code coverage, and the result was achieved.

File	index.js
Statements	100% (56/56)
Branches	100% (42/42)
Functions	100% (5/5)
Lines	100% (47/47)