## Part I

Basket 0: This is buggy. Every method in this implementation will give zero, which does not make sense.

Basket 1: This is buggy. For countItem method it used == instead of .equals, thus if we have an item with different reference but same content, it will not be counted to the total count. Also it did not consider the case of null.

Basket 2: This is ok. It is possible to have null value in the list, thus if there are multiple null values in the basket, it will return zero no matter how much nulls, but this is acceptable since null does not give us any information.

Basket 3: This is buggy, for totalCost method, it only take the second item in the list and use its price again and again, which will give a total price with multiple of the price of the second item, thus this does not make sense.

Basket 4: This is buggy, since the total length of the basket is only 10, which means if we add more than 10 things, it will break.

Basket 5: This is buggy, for countItem method, it returns when the item in the list equals to the item we want to count, thus if there are more this item later in the list, we cannot get those items counted.

Basket 6: This is ok, for removeFromBasket method, if the null exist in the list, then treat it is not there is null means nothing.

Basket 7: This is buggy, for totalCost method, since the index of the list starts at zero, if get(i), there will be one left and not counted into the total cost.

Basket 8: This is ok, for removeAllFromBasket method, it just write the opposite output compare to other baskets, it just will return false after it removes all the items we want to remove.

Basket 9: This is ok, although this does not allow you add null value into the list, for other method it did take care of the case of null, so it is fine.

Basket 10: This is buggy, for removeFromBasket method, the index could be zero, so when it minus one, it will give IndexOutOfBound exception.

Basket 11: This is buggy, for removeFromBasket method, it does not properly remove the item from the list, thus when using the totalCost method, it does not give the correct number.

Basket 12: This is buggy, for countItem method, it uses == instead of .equals thus it cannot count the item correctly.

## Part II

Basket2, Basket4, Basket6

Basket4 has the different addALot result than 2 and 6, there is a IndexOutOfBound exception from Basket4 since the max length of the basket is only 10, thus when we add 100 items into the list, it breaks. Since Basket4 and 2’s count method does take care of the null value, and it is not being considered as a value that take space in the list, thus the outcome will not match with Basket6 because Basket6 did not choose to neglect the null value. Because of the way Basket4 is created, there are null values at the beginning, thus when we are using the totalCost method, it will give a NullPointerException. In Basket2, the countItem method did not use .equals, thus when an item with the same content but different reference is added to the list, the countItem will not recognize it. Also for Basket6, the remove method functions correctly but returns an opposite value compare to other Baskets, thus give a different test result.