Checkout Kata

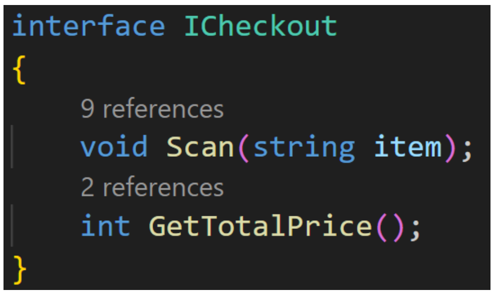
Implement the code for a checkout system that handles pricing schemes such as “pineapples cost 50, three pineapples cost 130”.

Implement the code for a supermarket checkout that calculates the total price of a number of items. In a normal supermarket, things are identified using Stock Keeping Units, or SKUs. In our store, we’ll use individual letters of the alphabet (A, B, C, and so on). Our goods are priced individually. In addition, some times are multi-priced: buy n of them and they’ll cost you y pence. For example, item A might cost 50 individually , but this week we have a special offer: buy three As and they’ll cost you 130. In fact the prices are:

SKU	**Unit Price**	**Special Price**
A	50	3 for 130
B	30	2 for 45
C	20	
D	15	

The checkout accepts items in any order, so that if we scan a B, an A, and another B, we’ll recognise the two Bs and price them at 45 (for a total price so far of 95). **The pricing changes frequently, so pricing should be independent to the checkout.**

The interface to the checkout could look like this:



```
interface ICheckout
{
    9 references
    void Scan(string item);
    2 references
    int GetTotalPrice();
}
```

Guidance

- Unit tests should cover behaviours of the system and edge cases.
- Making small commits and showing your commit history is recommended so that we can see your approach. For example, if you're using Test Driven Development then show us.
- Keep it simple and take as much time as you need, the aim is to demonstrate your approach.
- You do not need to deploy the application.
- There is no need to fork our repo. Submit a link to your GitHub repository via email.