# **UNiDAYS Discount Challenge**

## **How to Build and Run my project**

My project is run on Visual Studio (the 2019 version) using C++. I have 3 files, the main C++ file (.cpp), the header file (.h) for the class and the class C++ file (.cpp). The main file is the only one which should need editing for testing. If loaded into an empty project, then they’ll all run concurrently.

There are 2 things that are available for change, the basket and the pricing rules.

The basket can be changed by either adding in, or taking away, lines of the add basket method. Just simply write example.AddToBasket(“Name of Item”); as shown in figure 1.

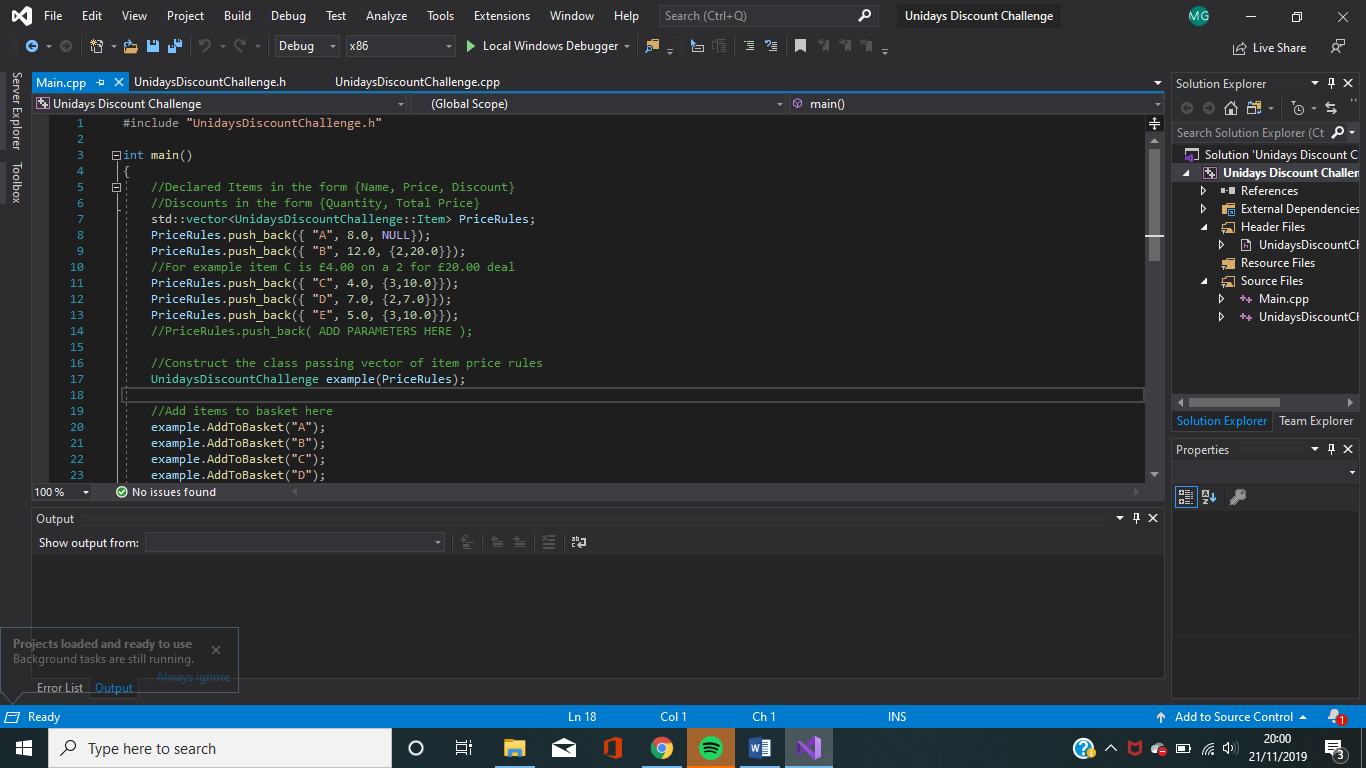
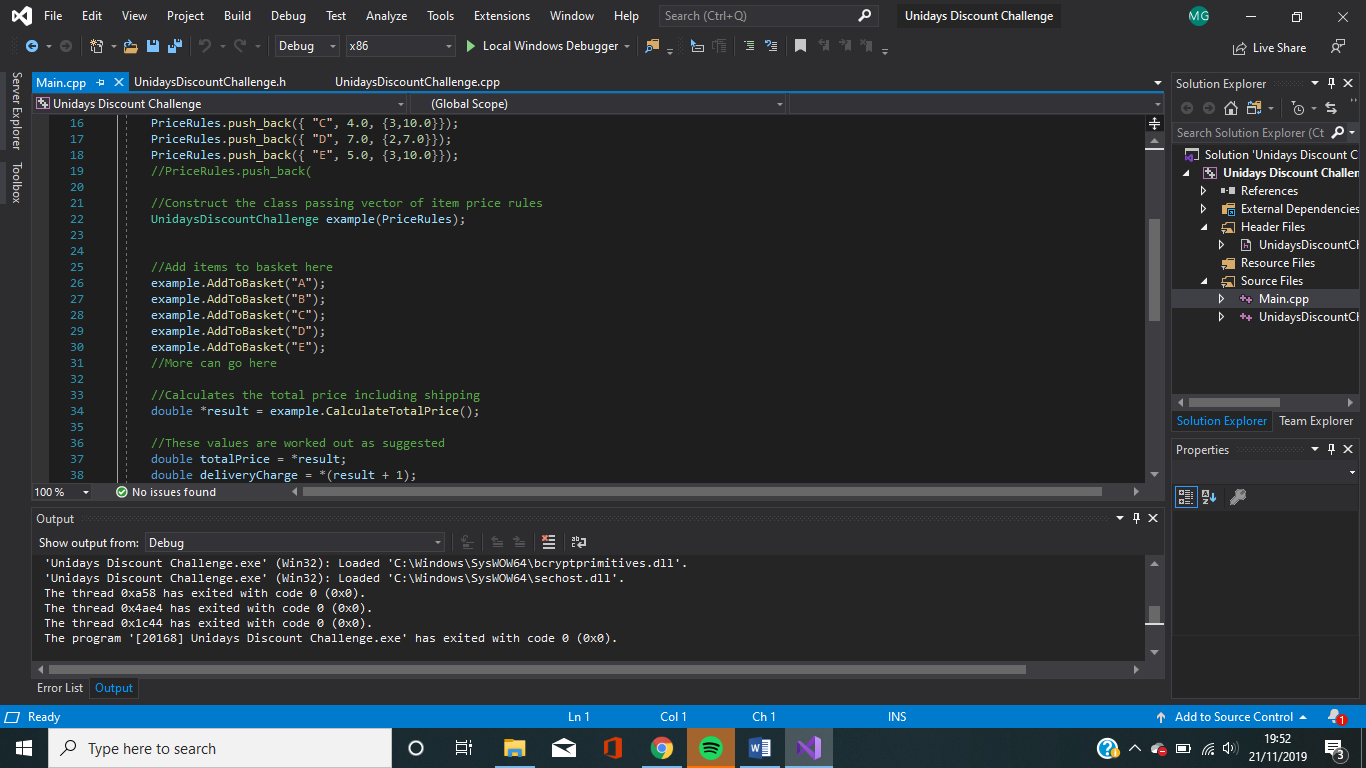


Figure AddToBasket method call Figure 2 Declaring Price Rules

Another changeable section is the pricing rules. There is the option to add additional pricing rules or perhaps less. You can add in a line, like the last line in figure 2 shows. The middle commented section demonstrates what the line of code below means in plain English and should give a good indication of how to add more price rules in.

## **How I went about solving the problem**

I started by getting a list of all the requirements. Here are my results:

* The Pricing Rule table must be followed, e.g. Item A is £8 and has no promotional deal and so on.
* Promotions must be followed. For example, there are 3 of item B, then a single promotion should happen and another of the full price added on to make £32
* An Item on 3 for 2 or 2 for 1 should have the last item free.
* Add on a delivery charge if the total cost is less than £50 and therefore if £50 or over then no delivery charge.
* There should only be 2 public methods, AddToBasket and CalculateTotalPrice.
* The class should be initialised with the pricing rules.
* From looking at the test cases, if there are no items, no delivery charge should be added.
* And finally, example variable names, which I have used.

All the requirements listed above have been met.

Where there is the option to add in or take away things from the system, it didn’t make sense to use an array. Therefore, I decided to use vectors as they are dynamic in memory and don’t require a size in the initialisation.

I’ve used a couple of strucs to group together some data. I used a struct called “Item”, which consisted of a name, price, discount and quantity. However, a discount was also hard to distinguish and therefore I also made a struct called “Discount”, which packaged together the number of items to make the deal and the deal price. This gave me all the information I needed to start adding items to a basket and calculating the total.

I have a constructor which iterates through the passed pricing rules and applies them to a class variable, such that the other methods in the class can use them.

How do the methods work?

* Add to Basket method: The way this method works is it looks at the item name passed through and then looks through the item rules to find a match. I’ve only used a linear search because in the example there is 5 elements and the time saved using a quicker search would be negligible. If a value is found the quantity will be incremented, then the loop will break.
* Calculate Total Price method: This will go through the items and check their quantities and compare them to the discounts using the modulo function. This will give the number of items we need to add on. I used integer division to see how many discounts to apply. Then from here we can multiply the number of deals to apply, by the deal price of each item and from here add the normal price of the item, multiplied by the remainder (if any). It will return a pointer to an array containing the total price and the delivery price to add.