MarketPrices

Write a container class in which bids and offers (also called orders) related to multiple products are stored.
The MarketPrices class is made up of different orderbooks. An orderbook is a set of orders associated to the same productId. Each orderbook can have bid and offer orders which are sorted based on the Price of the order.
Bids have to be ordered based on the Price in descending order while Offers have to be ordered based on the Price in ascending order.
It is possible to print the depth on the standard output using the print method.

```
class MarketPrices {
public:
        /* This function is called to enter a new Bid or Offer in the system. The order has to be
           stored in the correct position of the orderbook according to the sort order
           ProductId uniquely identifies the product on which the order is entered.
           OrderId uniquely identifies the order
           BidOrOffer indicates whether the order is a bid or an offer. 1 indicates it is a Bid,
           2 otherwise
           Returns 0 in case of errors */
        virtual int OnOrderAdd(char *productId, char *OrderId, int BidOrOffer, int Price) = 0;

        /* This function is called to delete an existing order present in the orderbook.
           ProductId uniquely identifies the product on which the order is entered.
           OrderId uniquely identifies the order to be deleted
           Returns 0 in case of errors */
        virtual int OnOrderDel(char *productId, char *OrderId) = 0;

        /* This function is called to print all the bid and offer orders present in the orderbook for all
           products.
           Returns 0 in case of errors */
        virtual int Print() = 0;

};
```

Examples:

```
OnOrderAdd(“product1”, “order1”, 1, 100)
OnOrderAdd(“product1”, “order2”, 1, 101)

OnOrderAdd(“product2”, “order3”, 1, 99)
OnOrderAdd(“product2”, “order4”, 2, 100)
```

Print has the following result:

```
product 1:
Buy:
order2 101
order1 100
Sell:

product 2:
Buy:
order3 99
Sell:
```

order4 100