2. Consider the following declaration that will be used to keep track of information about items in a grocery store . Each item is identified by a unique one-word name and has an associated price, size, and category.

```
class GroceryStore
{
  public:
    GroceryStore();

    // modifier

    void SetPrice(const apstring & itemName, double price);
       // changes the price of item associated with itemName

    // accessors

    double GetPrice(const apstring & itemName) const;
       // returns the price of this item

    int GetSize(const apstring & itemName) const;
       // returns the size (in ounces) of this item

    apvector<apstring> GetItems(char category) const;
       // returns a vector (possibly empty) of the names of all
       // items in the specified category

    // ... other public and private members not shown
};
```

(a) You will write free function ChangePrices, which is described as follows. ChangePrices reads item names and prices from input and changes the prices of the corresponding items in store to the new prices.

For example, assume store contains the following items.

Name	**Price**	**Size (in ounces)**	**Category**
avocado	1.68	8	P
milk	1.92	64	D
chicken	4.48	64	M
broccoli	1.92	16	P
yogurt	0.96	16	D
spinach	1.76	16	P
cornedbeef	6.72	48	M
porkchops	2.24	32	M

Assume that the stream input contains the following data.

```
cornedbeef  7.99
yogurt       .75
milk        1.25
broccoli     .98
```

The call ChangePrices(store, input) will change the prices of cornedbeef, yogurt, milk, and broccoli to the corresponding new prices.

In writing ChangePrices, you may call any of the public member functions of the GroceryStore class. Assume the member functions work as specified.

Complete free function ChangePrices below.

```
void ChangePrices(GroceryStore & store, istream & input)
// precondition:  input is open for reading;
//                each line consists of a valid one word item name
//                and a valid price
// postcondition: changes the prices of items in store using names and
//                new prices from input
```

GO ON TO THE NEXT PAGE.

(b) The unit price of an item is the price per ounce. The table below is repeated from part (a) for your convenience.

Name	**Price**	**Size (in ounces)**	**Category**
avocado	1.68	8	P
milk	1.92	64	D
chicken	4.48	64	M
broccoli	1.92	16	P
yogurt	0.96	16	D
spinach	1.76	16	P
cornedbeef	6.72	48	M
porkchops	2.24	32	M

The unit price of avocado is 1.68 divided by 8, which equals 0.21, and the unit price of spinach is 1.76 divided by 16, which equals 0.11.

You will write free function `BargainItem`, which is described as follows. `BargainItem` returns the name of an item whose unit price is the lowest in the specified category. If there is more than one item with the lowest unit price, any one of these items may be returned. If there are no items in the category, `BargainItem` returns `"none"`.

For example, consider the items and prices listed in the table above. Using this table, the results of three calls to `BargainItem` are shown below.

Function call	Returned value
`BargainItem(store, ’P’)`	`spinach`
`BargainItem(store, ’M’)`	`chicken` or `porkchops`
`BargainItem(store, ’B’)`	`none`

In writing `BargainItem`, you may call any of the public member functions of the `GroceryStore` class. Assume that the member functions work as specified.

Complete free function `BargainItem` below.

```
apstring BargainItem(const GroceryStore & store, char category)
// postcondition: returns the name of an item whose unit price
//                is the lowest in the specified category;
//                if no items in the specified category, returns "none"
```