The materials included in these files are intended for use by AP teachers for course and exam preparation in the classroom; permission for any other use must be sought from the Advanced Placement Program®. Teachers may reproduce them, in whole or in part, in limited quantities, for face-to-face teaching purposes but may not mass distribute the materials, electronically or otherwise. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here. This permission does not apply to any third-party copyrights contained herein.
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.

```c
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
{
  string name;
  double price;
  while (input >> name >> price)
  {
    store.SetPrice(name, price);
  }
}
```

GO ON TO THE NEXT PAGE
Complete free function BargainItem below.

```cpp
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"
{
    double price = 0, low = 0;
    apvector <apstring> list = store.GetItem(category);
    apstring lowest;
    if (list.length() == 0)
        return "none";
    for (int i = 0; i < list.length(); i++)
    {
        price = store.Price(list[i]) / store.Size(list[i]);
        if (price < low || low == 0)
        {
            low = price;
            lowest = list[i];
        }
    }
    return lowest;
}
```

GO ON TO THE NEXT PAGE.
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.

```cpp
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
{
    while (input >> item_name >> price)
    
        store.setPrice(item_name, price)
```
Complete free function BargainItem below.

```cpp
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"
{
    int low = 3200, count;
    apstring n;
    apvector<apstring> list = get_items(category);
    for (count = 0; count < list.length(); count++)
    {
        if (store.getPrice(list[count]) / store getSize(list[count]) < low)
        {
            low = store.getPrice(list[count]) / store getSize(list[count]);
            n = list[count];
        }
    }
    return n;
}
```

GO ON TO THE NEXT PAGE.
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.

```c
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
{
    string current_name;
    double current_price;

    while (input >> current_name)
    {
        input >> current_price;
        store.setPrice(current_name, current_price);
    }

    return;
}
```

GO ON TO THE NEXT PAGE.
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"
{

  apstring current_bargain("none");
  apvector <apstring> category_items = store.items(category);
  double lowest, current;

  if (category_items.size())
    lowest = store.items(category).front();
  else
    return current_bargain;

  for (x=1; x<category_items.size(); x++)
    if (current = store.items(category)[x])
      if (current < lowest)
        lowest = current;

  current_bargain = store.items(category)[x];

  return current_bargain;
}