```
#include <iomanip>
#include <iostream>
#include <vector>
#include <string>
using namespace std;

class GroceryItem {
private:
   string name;
   int price;
   int quantity;

public:
   // The constructor
   GroceryItem(const string & name, int price, int quantity)
      : name(name), price(price), quantity(quantity) {}

   // The destructor
   virtual ~GroceryItem() {}

   // The copy constructor
   GroceryItem(const GroceryItem & other)
      : name(other.name), price(other.price), quantity(other.quantity) {}

   // The copy assignment operator
   GroceryItem & operator = (const GroceryItem & other) {
      if (this != &other) {
         name = other.name;
         price = other.price;
```

```
        quantity = other.quantity;
    }
    return *this;
  }

  int calculateSum() const {
    return price * quantity;
  }

  void display() const {
    cout << left << setw(10) << name << "PHP " << setw(4) << price << "x" << setw(2) << quantity <<
endl;
  }

  const string & getName() const {
    return name;
  }
};

// Class for Fruit
class Fruit : public GroceryItem {
public:
  Fruit(const string & name, int price, int quantity)
    : GroceryItem(name, price, quantity) {}

  ~Fruit() {}
};

// Class for Vegetable
```

```
class Vegetable : public GroceryItem {
public:
   Vegetable(const string & name, int price, int quantity)
      : GroceryItem(name, price, quantity) {}

   ~Vegetable() {}
};

// Function to display all items in the list
void displayGroceryList(const vector <GroceryItem*> & groceryList) {
   for (const auto & item : groceryList) {
      item -> display();
   }
}

// Function to calculate the total sum of all items in the list
int totalSum(const vector <GroceryItem*> & groceryList) {
   int sum = 0;
   for (const auto & item : groceryList) {
      sum += item -> calculateSum();
   }
   return sum;
}

// Function to delete an item from the list
void deleteItem(vector <GroceryItem*> & groceryList, const string& itemName) {
   for (auto it = groceryList.begin(); it != groceryList.end(); ++it) {
      if ((*it) -> getName() == itemName) {
         delete *it;
```

```
            groceryList.erase(it);
            break;
        }
    }
}

int main() {
    // Problem 2: Create an array GroceryList
    vector <GroceryItem*> groceryList = {
        new Fruit("Apple", 10, 7),
        new Fruit("Banana", 10, 8),
        new Vegetable("Broccoli", 60, 12),
        new Vegetable("Lettuce", 50, 10)
    };
    cout << "Grocery List:\n";
    displayGroceryList(groceryList);

    // Problem 3: Calculate the total sum
    cout << "\nTotal Sum: PHP " << totalSum(groceryList) << endl;

    // Problem 4: Delete Lettuce and deallocate memory
    deleteItem(groceryList, "Lettuce");
    cout << "\nGrocery List (After the deletion of Lettuce):\n";
    displayGroceryList(groceryList);
    cout << "\nTotal Sum: PHP " << totalSum(groceryList) << endl;
    for (auto & item : groceryList) {
        delete item;
    }
```

```
    return 0;
}
```