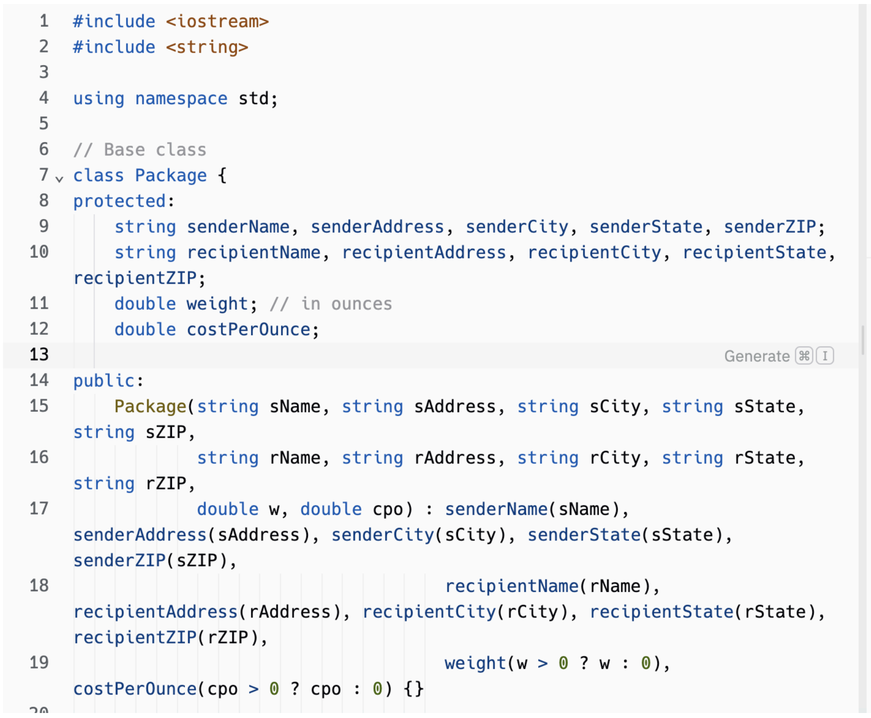
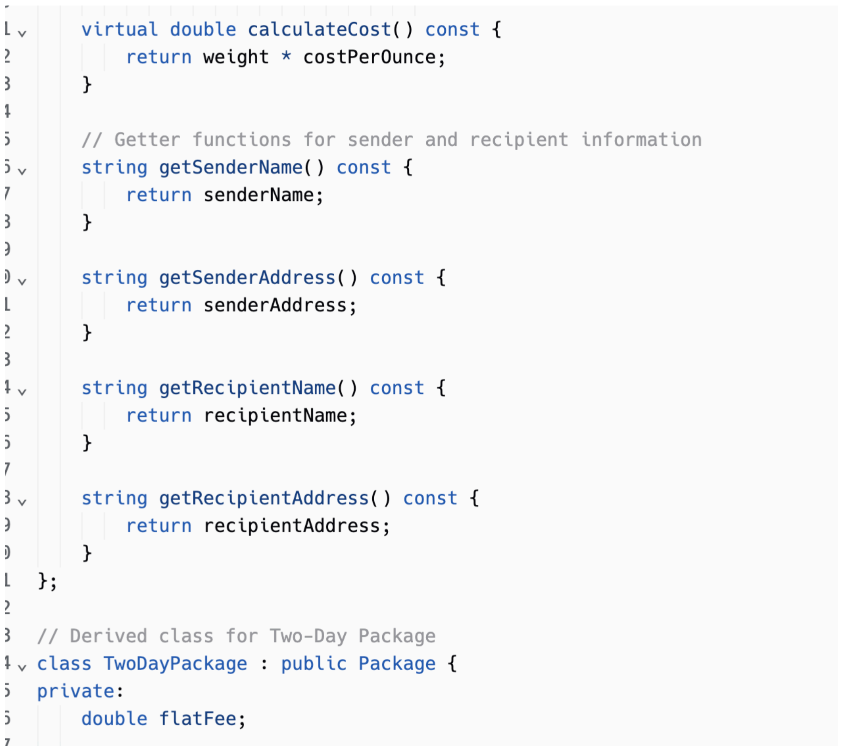
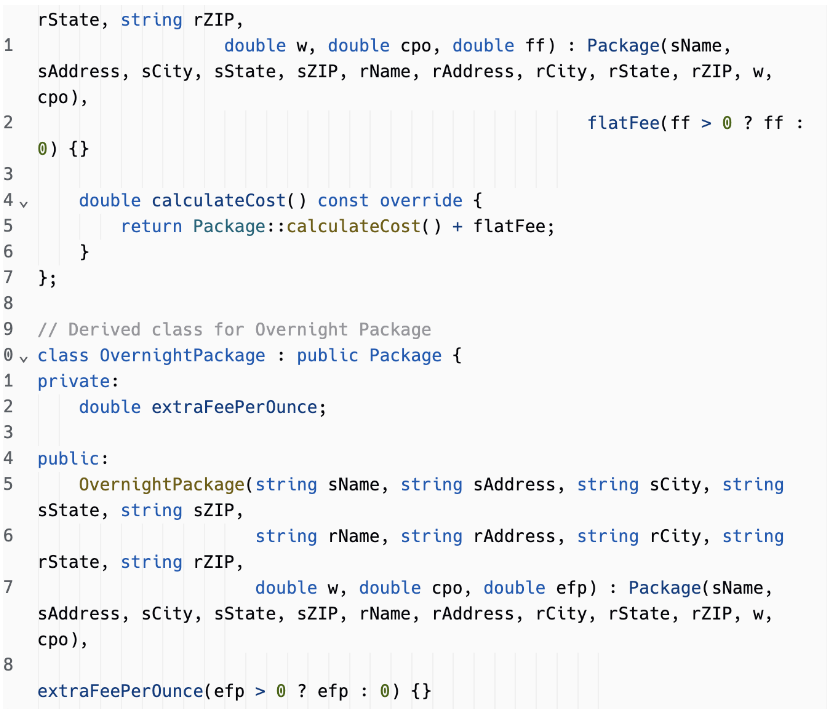
Q.no.1)



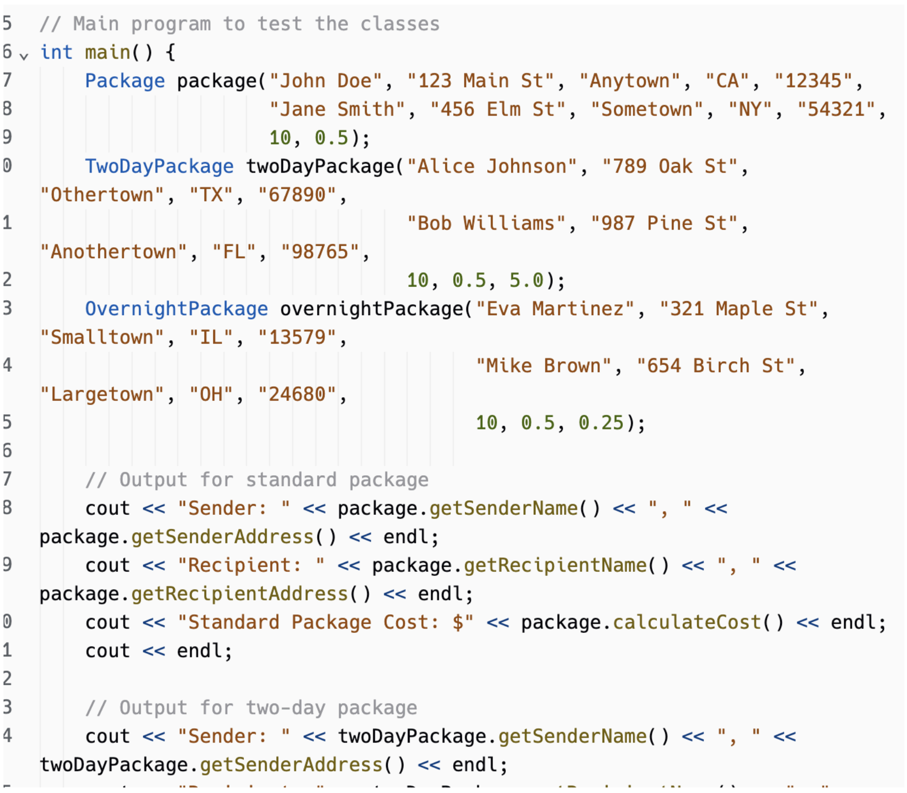
```cpp
#include <iostream>
#include <string>

using namespace std;

// Base class
class Package {
protected:
    string senderName, senderAddress, senderCity, senderState, senderZIP;
    string recipientName, recipientAddress, recipientCity, recipientState, recipientZIP;
    double weight; // in ounces
    double costPerOunce;

public:
    Package(string sName, string sAddress, string sCity, string sState, string sZIP,
            string rName, string rAddress, string rCity, string rState, string rZIP,
            double w, double cpo) : senderName(sName), senderAddress(sAddress), senderCity(sCity), senderState(sState), senderZIP(sZIP),
                                    recipientName(rName), recipientAddress(rAddress), recipientCity(rCity), recipientState(rState), recipientZIP(rZIP),
                                    weight(w > 0 ? w : 0), costPerOunce(cpo > 0 ? cpo : 0) {}
```



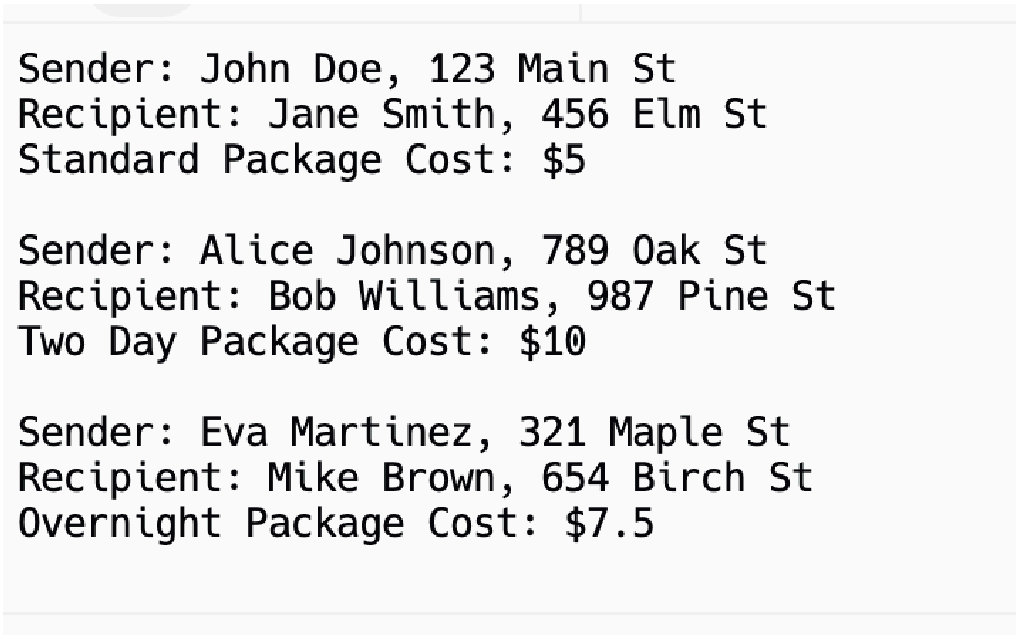
```cpp
    virtual double calculateCost() const {
        return weight * costPerOunce;
    }

    // Getter functions for sender and recipient information
    string getSenderName() const {
        return senderName;
    }

    string getSenderAddress() const {
        return senderAddress;
    }

    string getRecipientName() const {
        return recipientName;
    }

    string getRecipientAddress() const {
        return recipientAddress;
    }
};

// Derived class for Two-Day Package
class TwoDayPackage : public Package {
private:
    double flatFee;
```



```cpp
rState, string rZIP,
                 double w, double cpo, double ff) : Package(sName, sAddress, sCity, sState, sZIP, rName, rAddress, rCity, rState, rZIP, w, cpo),
                                                   flatFee(ff > 0 ? ff : 0) {}

    double calculateCost() const override {
        return Package::calculateCost() + flatFee;
    }
};

// Derived class for Overnight Package
class OvernightPackage : public Package {
private:
    double extraFeePerOunce;

public:
    OvernightPackage(string sName, string sAddress, string sCity, string sState, string sZIP,
                     string rName, string rAddress, string rCity, string rState, string rZIP,
                     double w, double cpo, double efp) : Package(sName, sAddress, sCity, sState, sZIP, rName, rAddress, rCity, rState, rZIP, w, cpo),
extraFeePerOunce(efp > 0 ? efp : 0) {}
```

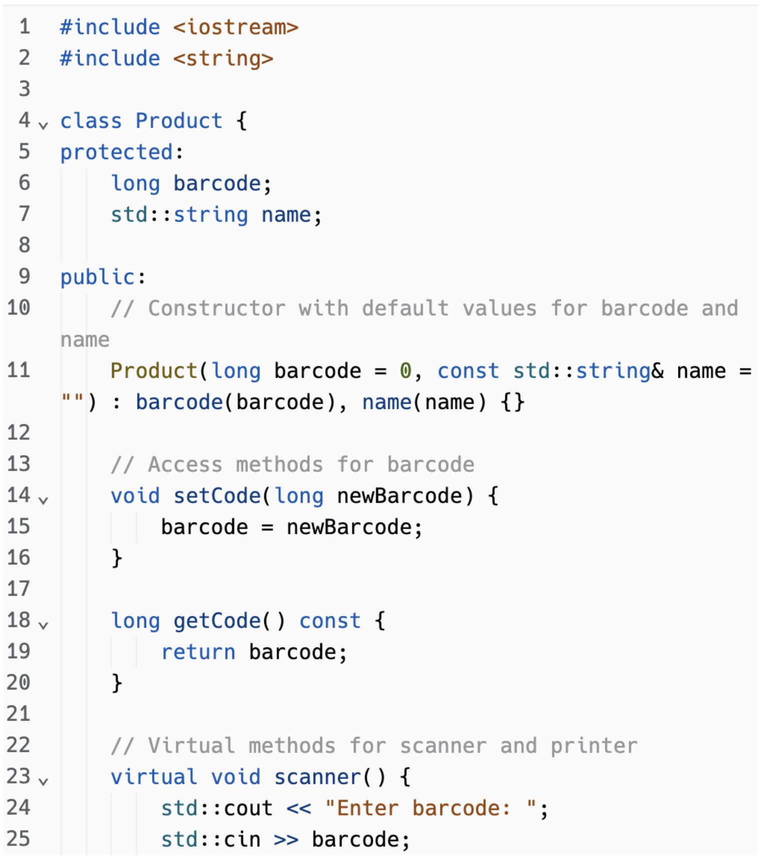


```cpp
// Main program to test the classes
int main() {
    Package package("John Doe", "123 Main St", "Anytown", "CA", "12345",
                    "Jane Smith", "456 Elm St", "Sometown", "NY", "54321",
                    10, 0.5);
    TwoDayPackage twoDayPackage("Alice Johnson", "789 Oak St", "Othertown", "TX", "67890",
                                "Bob Williams", "987 Pine St", "Anothertown", "FL", "98765",
                                10, 0.5, 5.0);
    OvernightPackage overnightPackage("Eva Martinez", "321 Maple St", "Smalltown", "IL", "13579",
                                      "Mike Brown", "654 Birch St", "Largetown", "OH", "24680",
                                      10, 0.5, 0.25);

    // Output for standard package
    cout << "Sender: " << package.getSenderName() << ", " << package.getSenderAddress() << endl;
    cout << "Recipient: " << package.getRecipientName() << ", " << package.getRecipientAddress() << endl;
    cout << "Standard Package Cost: $" << package.calculateCost() << endl;
    cout << endl;

    // Output for two-day package
    cout << "Sender: " << twoDayPackage.getSenderName() << ", " << twoDayPackage.getSenderAddress() << endl;
```

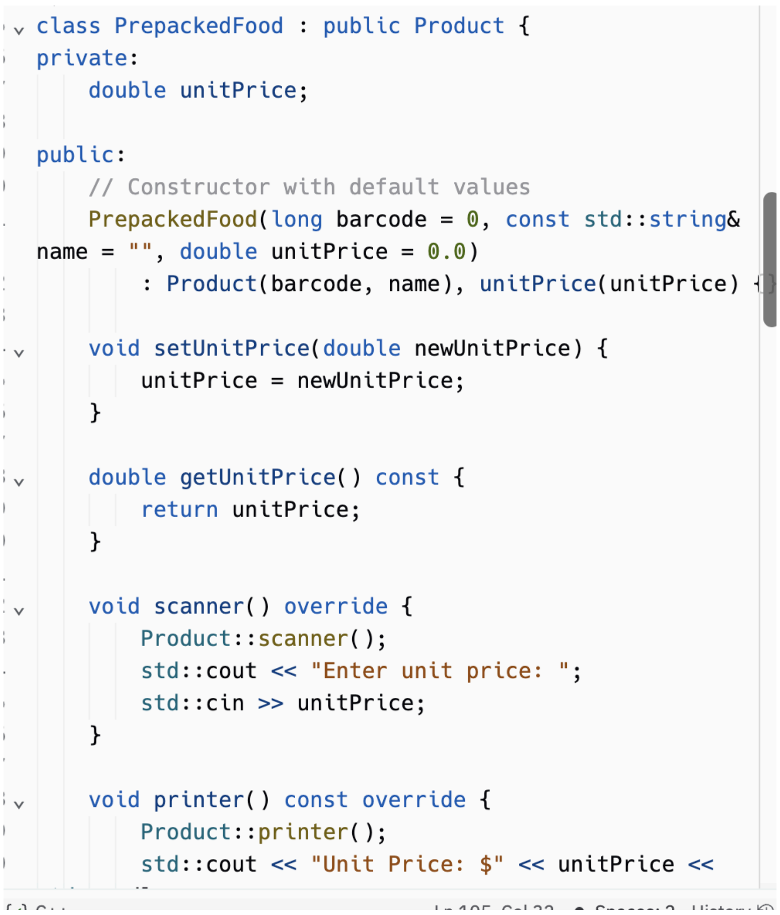


```
Sender: John Doe, 123 Main St
Recipient: Jane Smith, 456 Elm St
Standard Package Cost: $5

Sender: Alice Johnson, 789 Oak St
Recipient: Bob Williams, 987 Pine St
Two Day Package Cost: $10

Sender: Eva Martinez, 321 Maple St
Recipient: Mike Brown, 654 Birch St
Overnight Package Cost: $7.5
```

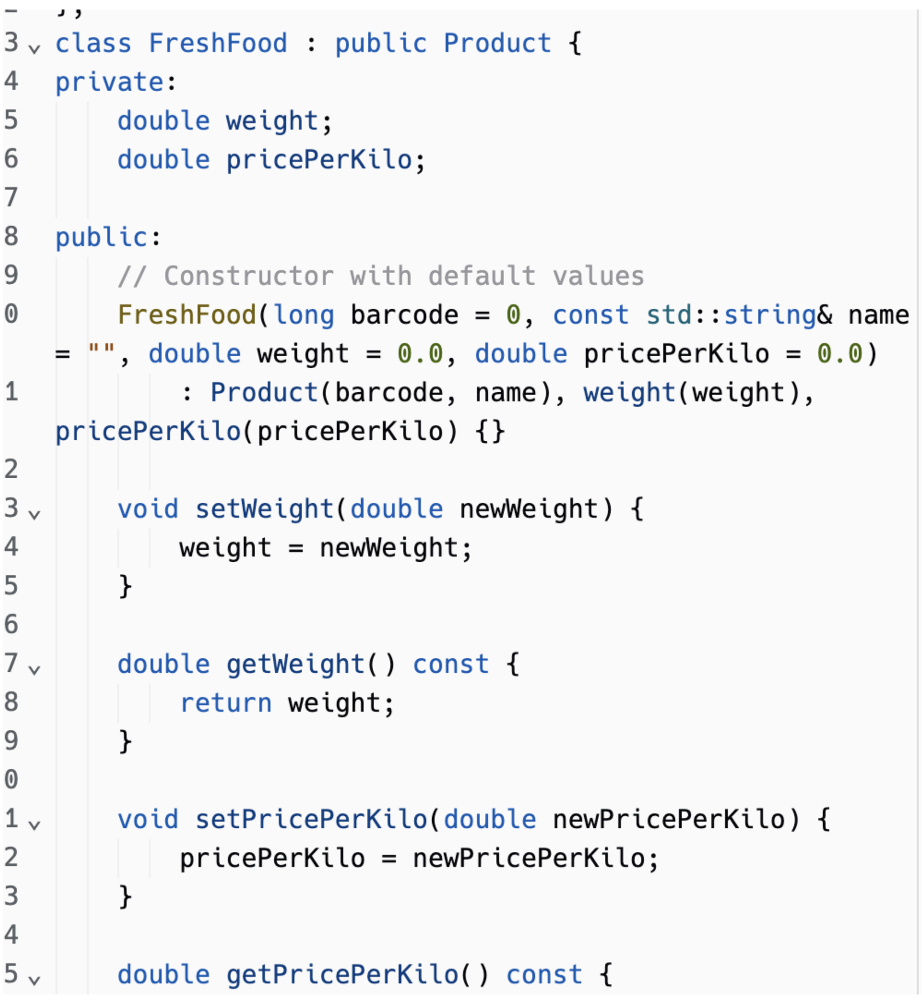
Q.no.2)



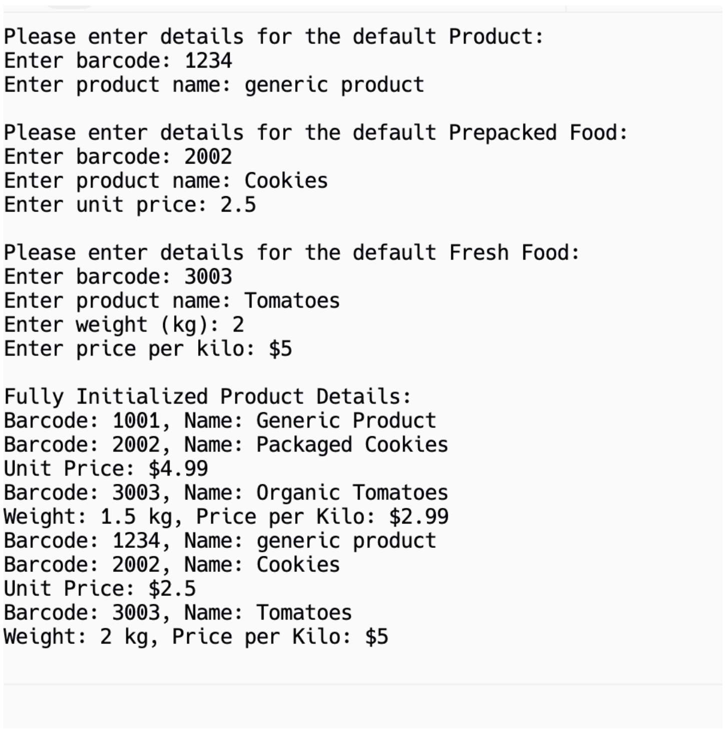
```cpp
#include <iostream>
#include <string>

class Product {
protected:
    long barcode;
    std::string name;

public:
    // Constructor with default values for barcode and name
    Product(long barcode = 0, const std::string& name = "") : barcode(barcode), name(name) {}

    // Access methods for barcode
    void setCode(long newBarcode) {
        barcode = newBarcode;
    }

    long getCode() const {
        return barcode;
    }

    // Virtual methods for scanner and printer
    virtual void scanner() {
        std::cout << "Enter barcode: ";
        std::cin >> barcode;
```



```cpp
class PrepackedFood : public Product {
private:
    double unitPrice;

public:
    // Constructor with default values
    PrepackedFood(long barcode = 0, const std::string& name = "", double unitPrice = 0.0)
        : Product(barcode, name), unitPrice(unitPrice) {}

    void setUnitPrice(double newUnitPrice) {
        unitPrice = newUnitPrice;
    }

    double getUnitPrice() const {
        return unitPrice;
    }

    void scanner() override {
        Product::scanner();
        std::cout << "Enter unit price: ";
        std::cin >> unitPrice;
    }

    void printer() const override {
        Product::printer();
        std::cout << "Unit Price: $" << unitPrice <<
```



```cpp
};
class FreshFood : public Product {
private:
    double weight;
    double pricePerKilo;

public:
    // Constructor with default values
    FreshFood(long barcode = 0, const std::string& name = "", double weight = 0.0, double pricePerKilo = 0.0)
        : Product(barcode, name), weight(weight), pricePerKilo(pricePerKilo) {}

    void setWeight(double newWeight) {
        weight = newWeight;
    }

    double getWeight() const {
        return weight;
    }

    void setPricePerKilo(double newPricePerKilo) {
        pricePerKilo = newPricePerKilo;
    }

    double getPricePerKilo() const {
```



```
Please enter details for the default Product:
Enter barcode: 1234
Enter product name: generic product

Please enter details for the default Prepacked Food:
Enter barcode: 2002
Enter product name: Cookies
Enter unit price: 2.5

Please enter details for the default Fresh Food:
Enter barcode: 3003
Enter product name: Tomatoes
Enter weight (kg): 2
Enter price per kilo: $5

Fully Initialized Product Details:
Barcode: 1001, Name: Generic Product
Barcode: 2002, Name: Packaged Cookies
Unit Price: $4.99
Barcode: 3003, Name: Organic Tomatoes
Weight: 1.5 kg, Price per Kilo: $2.99
Barcode: 1234, Name: generic product
Barcode: 2002, Name: Cookies
Unit Price: $2.5
Barcode: 3003, Name: Tomatoes
Weight: 2 kg, Price per Kilo: $5
```