Lớp: CS112.P11.CTTN
Nhóm: 14
Sinh viên: Lê Nguyễn Anh Khoa. MSSV: 23520742
Sinh viên: Cáp Kim Hải Anh. MSSV: 23520036

BÀI TẬP
Kiểm tra tính đúng đắn và hiệu năng của chương trình bằng bộ test

Bài 1:

1. Viết mã giả cho bài toán:

```
Function TinhChiPhi(Order order)
    totalCost = 0
    totalCostWithoutDiscount = 0
    // For each product in order.Products
        productCost = product.Price * product.Quantity
        totalCostWithoutDiscount += productCost

        if product.DiscountPercentage > 0
            discountAmount = productCost * (product.DiscountPercentage / 100)
            productCost -= discountAmount

        totalCost += productCost

    if totalCostWithoutDiscount < 1000000
        totalCost += order.ShippingFee

    if order.IsRegularCustomer
        regularCustomerDiscount = totalCost * 0.1
        totalCost -= regularCustomerDiscount
    return totalCost

Function TinhTongGiaThanhKhongGiamGia(Order order)
    total = 0
```

```
    For each product in order.Products
        total += product.Price * product.Quantity
    return total


Function TinhTongGiaThanhCoGiamGia(Order order)
    total = 0
    For each product in order.Products
        productCost = product.Price * product.Quantity
        if product.DiscountPercentage > 0
            discountAmount = productCost * (product.DiscountPercentage / 100)
            productCost -= discountAmount
        total += productCost
    return total
```

2. Dựa vào mã giả, áp dụng Unit test, White box test, Black box test cho các phần nào của bài toán. Nếu áp dụng White box test/ Black box test thì ta cần sinh các test có đặc điểm như thế nào?

a) Unit Testing:

Áp dụng cho các hàm riêng lẻ: TinhChiPhi, TinhTongGiaThanhKhongGiamGia, TinhTongGiaThanhCoGiamGia.

Đặc điểm của các test case:

Kiểm tra các trường hợp cơ bản và biên.

Kiểm tra các trường hợp có và không có giảm giá sản phẩm.

Kiểm tra các trường hợp có và không có phí vận chuyển.

Kiểm tra các trường hợp khách hàng thường xuyên và không thường xuyên.

b) White Box Testing:

Áp dụng cho tất cả các hàm, đặc biệt là hàm TinhChiPhi.

Đặc điểm của các test case:

Đảm bảo tất cả các nhánh của câu lệnh điều kiện được thực thi.

Kiểm tra các vòng lặp với số lượng sản phẩm khác nhau (0, 1, nhiều).

Kiểm tra các trường hợp biên của các điều kiện (ví dụ: totalCostWithoutDiscount = 999999, 1000000, 1000001).

c) Black Box Testing:

Áp dụng cho toàn bộ hệ thống tính toán đơn hàng.

Đặc điểm của các test case:

Kiểm tra các trường hợp đầu vào hợp lệ và không hợp lệ.

Kiểm tra các tình huống thực tế khác nhau (đơn hàng nhỏ, lớn, nhiều sản phẩm, ít sản phẩm).

Kiểm tra các trường hợp đặc biệt (giảm giá 100%, số lượng sản phẩm là 0, giá sản phẩm là 0).

*** Ví dụ về các test case cụ thể:**

a) Unit Test cho hàm TinhTongGiaThanhCoGiamGia:

Test case 1: Đơn hàng không có sản phẩm

Test case 2: Đơn hàng có 1 sản phẩm không giảm giá

Test case 3: Đơn hàng có 1 sản phẩm có giảm giá 50%

Test case 4: Đơn hàng có nhiều sản phẩm với các mức giảm giá khác nhau

b) White Box Test cho hàm TinhChiPhi:

Test case 1: Đơn hàng có giá trị 999,999 đồng (kiểm tra biên của điều kiện miễn phí vận chuyển)

Test case 2: Đơn hàng có giá trị 1,000,000 đồng (kiểm tra biên của điều kiện miễn phí vận chuyển)

Test case 3: Đơn hàng của khách hàng thường xuyên với giá trị lớn hơn 1,000,000 đồng

Test case 4: Đơn hàng của khách hàng không thường xuyên với giá trị nhỏ hơn 1,000,000 đồng

c) Black Box Test cho toàn bộ hệ thống:

Test case 1: Đơn hàng thông thường (nhiều sản phẩm, có giảm giá, khách hàng không thường xuyên)

Test case 2: Đơn hàng lớn của khách hàng thường xuyên (nhiều sản phẩm, có giảm giá, miễn phí vận chuyển)

Test case 3: Đơn hàng nhỏ (1 sản phẩm, không giảm giá, có phí vận chuyển)

Test case 4: Đơn hàng với sản phẩm giảm giá 100%

Test case 5: Đơn hàng không có sản phẩm

Các test case này sẽ giúp đảm bảo tính chính xác và độ tin cậy của hệ thống tính toán đơn hàng.

Bài 2: Dãy con có tổng lớn nhất

1. Viết code trâu có độ phức tạo $O(n^2)$ hoặc $O(n^3)$

```cpp
#include <bits/stdc++.h>
using namespace std;
template <class T> inline bool maxx(T &A,T B){return A < B ? (A = B,1) : 0;}
const int N = 1e5 + 5;
int n, a[N], res = -1e9;
signed main(){
    cin >> n;
    for(int i = 1; i <= n; ++i){
        cin >> a[i];
        a[i] += a[i-1];
        for(int j = 0; j < i; ++j)
            maxx(res, a[i] - a[j]);
    }
    cout << res;
}

```

2. Viết code full có độ phức tạp O(n) hoặc O(n.logn)

```cpp
#include <bits/stdc++.h>
using namespace std;
template <class T> inline bool maxx(T &A,T B){return A < B ? (A = B,1) : 0;}
template <class T> inline bool minn(T &A,T B){return A > B ? (A = B,1) : 0;}
const int N = 1e5 + 5;
int n, a[N], res = -1e9, Min;
signed main(){
    cin >> n;
    for(int i = 1; i <= n; ++i){
        cin >> a[i];
        a[i] += a[i-1];
        maxx(res, a[i] - Min);
        minn(Min, a[i]);
    }
    cout << res;
}

```

3. Viết trình sinh và so test giữa "code trâu" và "code full"

Mã giải:

Function GenerateTestCases():

    testcase = []

    for i = 1 to num:

        arr = random(size, minValue, maxValue)

        testcase.append(arr)

    return testcase

4. Trình bày các trường hợp đặc biệt về test case của bài toán:

- Tất cả các số là số âm, cần lấy giá trị nhỏ nhất là số lớn nhất
- Tất cả các số là số dương, dãy con có tổng lớn nhất là toàn bộ dãy
- Dãy chỉ có một phần tử: kết quả là chính nó
- Dãy có các phần tử âm và dương đan xen: kiểm tra tính ổn định của thuật toán
- Dãy có giá trị rất lớn: tổng có thể tiệm cận đến giới hạn int