TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM

KHOA KHOA HỌC MÁY TÍNH

BÀI TẬP KIỂM TRA TÍNH ĐÚNG ĐẮN & HIỆU NĂNG CỦA CHƯƠNG TRÌNH BẰNG BỘ TEST

Môn học: CS112.P11.KHTN – Phân tích và thiết kế thuật toán

Thực hiện bởi nhóm 4, bao gồm:

1. Phan Nhật Tân 23521405
2. Nguyễn Huy Phước 23521234

MỤC LỤC

1. Bài 1:

1.1. Mã giả (Pseudo code) cho bài toán:

```
# Định nghĩa cấu trúc dữ liệu cho Sản phẩm
Product:
  - Price (giá gốc)
  - Quantity (số lượng)
  - DiscountPercentage (phần trăm giảm giá)

# Định nghĩa cấu trúc dữ liệu cho Đơn hàng
Order:
  - Products (danh sách sản phẩm)
  - IsRegularCustomer (khách hàng thường xuyên hay không)
  - ShippingFee (phí vận chuyển)

# Hàm tính tổng giá trị trước khi áp dụng giảm giá
Function CalculateTotalPriceBeforeDiscount(Order order):
  totalPrice = 0
  For Each product in order.Products:
    totalPrice += product.Price * product.Quantity
  Return totalPrice

# Hàm tính tổng giá trị sau khi áp dụng giảm giá sản phẩm
Function CalculateTotalPriceWithDiscount(Order order):
  totalPrice = 0
  For Each product in order.Products:
    discountAmount = product.Price * product.Quantity * product.DiscountPercentage / 100
    totalPrice += (product.Price * product.Quantity) - discountAmount
  Return totalPrice

# Hàm tính tổng chi phí đơn hàng (gồm giảm giá, chiết khấu, và vận chuyển)
Function TinhChiPhi(Order order):
  totalPriceWithDiscount = CalculateTotalPriceWithDiscount(order)

  # Áp dụng chiết khấu cho khách hàng thường xuyên
  If order.IsRegularCustomer:
    totalPriceWithDiscount *= 0.9

  # Kiểm tra và áp dụng phí vận chuyển
  If CalculateTotalPriceBeforeDiscount(order) < 1000000:
    totalPriceWithDiscount += order.ShippingFee

  Return totalPriceWithDiscount

```

1.2. Áp dụng các phương pháp kiểm thử:

Unit Test:

- **Phần cần kiểm thử:** Các hàm tính toán riêng lẻ (**CalculateTotalPriceBeforeDiscount**, **CalculateTotalPriceWithDiscount**, và logic áp dụng chiết khấu/vận chuyển trong **TinhChiPhi**).
- **Đặc điểm test cases:**
    - **Input/Expected Output:**
        - **CalculateTotalPriceBeforeDiscount**:
            - Input: Danh sách sản phẩm (giá, số lượng).

        - Expected Output: Tổng giá trị trước giảm giá.
    - **CalculateTotalPriceWithDiscount**:
        - Input: Danh sách sản phẩm (giá, số lượng, giảm giá).
        - Expected Output: Tổng giá trị sau giảm giá.
    - **TinhChiPhi** (chỉ logic chiết khấu và vận chuyển):
        - Input: Trạng thái khách hàng, tổng giá trị đơn hàng trước/với giảm giá.
        - Expected Output: Tổng chi phí cuối cùng.
- **Test Scenarios:**
    - Giá trị dương, âm, và 0 cho giá sản phẩm, số lượng, giảm giá.
    - Trạng thái khách hàng thường xuyên và không thường xuyên.
    - Tổng giá trị đơn hàng lớn hơn, nhỏ hơn, và bằng 1 triệu.

White Box Test:

- **Phần cần kiểm thử:** Luồng quyết định và logic áp dụng giảm giá, chiết khấu, và phí vận chuyển trong **TinhChiPhi**.
- **Đặc điểm test cases:**
    - **Coverage mục tiêu:**
        - **Decision Coverage:** Kiểm tra tất cả nhánh (if/else) liên quan đến giảm giá sản phẩm, chiết khấu khách hàng, và phí vận chuyển được thực thi.
        - **Condition Coverage:** Mỗi điều kiện trong quyết định (ví dụ: **IsRegularCustomer**, tổng giá trị đơn hàng so với 1 triệu) được kiểm tra với cả giá trị True và False.
    - **Test Scenarios:**
        - Đơn hàng với sản phẩm có giảm giá và không giảm giá.
        - Khách hàng thường xuyên và không thường xuyên với tổng giá trị đơn hàng khác nhau (dưới, trên, bằng 1 triệu).

Black Box Test:

- **Phần cần kiểm thử:** Hành vi tổng thể của hàm **TinhChiPhi** đối với các loại đầu vào hợp lệ và không hợp lệ.
- **Đặc điểm test cases:**
    - **Equivalence Partitioning:**
        - Đầu vào hợp lệ (tổng giá trị đơn hàng, trạng thái khách hàng, giá sản phẩm, số lượng, giảm giá).
        - Đầu vào không hợp lệ (giá trị âm, văn bản thay cho số, v.v.).
    - **Boundary Value Analysis:**
        - Tổng giá trị đơn hàng ngay tại, dưới, và trên ngưỡng 1 triệu.
        - Giảm giá sản phẩm tại, dưới, và trên 100%.
    - **Test Scenarios:**
        - Đơn hàng trống.
        - Đơn hàng với nhiều sản phẩm có giảm giá khác nhau.
        - Thay đổi trạng thái khách hàng và quan sát sự khác biệt trong tổng chi phí.

2. Bài 2:

2.1. Code trâu:

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+10;
int n, sum, ans, A[N];
int main(){
        ios::sync_with_stdio(0);cin.tie(0);
        freopen("file.inp","r",stdin);
        freopen("file.ans","w",stdout);
        cin>>n;
        for(int i=1; i<=n; i++) cin>>A[i];
        ans=-2e9;
         //O(N^3)
        for(int i=1; i<=n; i++){
            for(int j=i; j<=n; j++){
                    sum=0;
                    for(int z=i; z<=j; z++) sum+=A[z];
                    ans=max(ans,sum);
            }
        }

        cout<<ans;


}
```

2.2. Code full:

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+10;
int n, sum, ans, A[N], pre_sum[N];
int main(){
        ios::sync_with_stdio(0);cin.tie(0);
        freopen("file.inp","r",stdin);
        freopen("file.out","w",stdout);

        cin>>n;
        for(int i=1; i<=n; i++) {
                cin>>A[i];
                pre_sum[i]=A[i]+pre_sum[i-1];
        }
        ans=-2e9;
         //O(N)
         int mi=0;
        for(int i=1; i<=n; i++){
             ans=max(ans,pre_sum[i]-mi);
             mi=min(mi,pre_sum[i]);
        }
        cout<<ans;


}
```

2.3. Trình sinh test:

```cpp
#include <bits/stdc++.h>
using namespace std;
mt19937 rd(chrono::steady_clock::now().time_since_epoch().count());

int get(int a,int b){
    if(a==b) return a;
    int t=rd();
    t=abs(t);
    return a+t%(b-a+1);
}

int main(){
    for(int test=1; test<=300; test++){
    srand(time(NULL));
    ofstream inp("file.inp");
     //viet code sinh test
     int n,x,q,l,r;
        n=get(1,100);
        inp<<n<<'\n';
        for(int i=1; i<=n; i++) inp<<get(-10000,10000)<<' ';
        inp<<'\n';

    inp.close();
    auto st=clock();

    system("trau_1.exe");
    system("ac_3.exe");
    double fi=clock();
    if(system("fc file.out file.ans")!=0){
        cout<<"wrong"<<'\n';
        return 0;
    }
    cout<<"OK"<<'\n';
     cout<<"Thoi giay chay: "<<(fi-st)/1000.0<<"s"<<'\n';


    }
    return 0;
}
```

2.4. Các trường hợp đặc biệt:

Tất cả phần tử trong dãy đều âm

- **Input: n > 1**, tất cả **ai < 0** (ví dụ: **-1, -2, -3,...**)
- **Expected Output:** Giá trị âm nhỏ nhất (tương ứng với phần tử âm đơn lẻ có giá trị tuyệt đối nhỏ nhất) hoặc 0 nếu giải thuật cho phép trả về 0 cho trường hợp này