Nguyễn Dương Ánh - 20210082

Bài thực hành số 5

Contents

[Bài 1 2](#_Toc140530261)

[Trước khi sửa 2](#_Toc140530262)

[Sau khi sửa 3](#_Toc140530263)

[Code 6](#_Toc140530264)

[Bài 2 7](#_Toc140530265)

[Trước khi sửa 7](#_Toc140530266)

[Sau khi sửa 9](#_Toc140530267)

[Code 9](#_Toc140530268)

[Bài 3 11](#_Toc140530269)

[Trước khi sửa 11](#_Toc140530270)

[Sau khi sửa 12](#_Toc140530271)

[Code 13](#_Toc140530272)

[Bài 4 15](#_Toc140530273)

[Bài 5 15](#_Toc140530274)

[Code 15](#_Toc140530275)

[Bài 6 16](#_Toc140530276)

[Code 16](#_Toc140530277)

[Bài 7 18](#_Toc140530278)

[Code 18](#_Toc140530279)

[Bài 8 20](#_Toc140530280)

[Trước khi sửa 20](#_Toc140530281)

[Sau khi sửa 21](#_Toc140530282)

[Code 21](#_Toc140530283)

[Bài 9 23](#_Toc140530284)

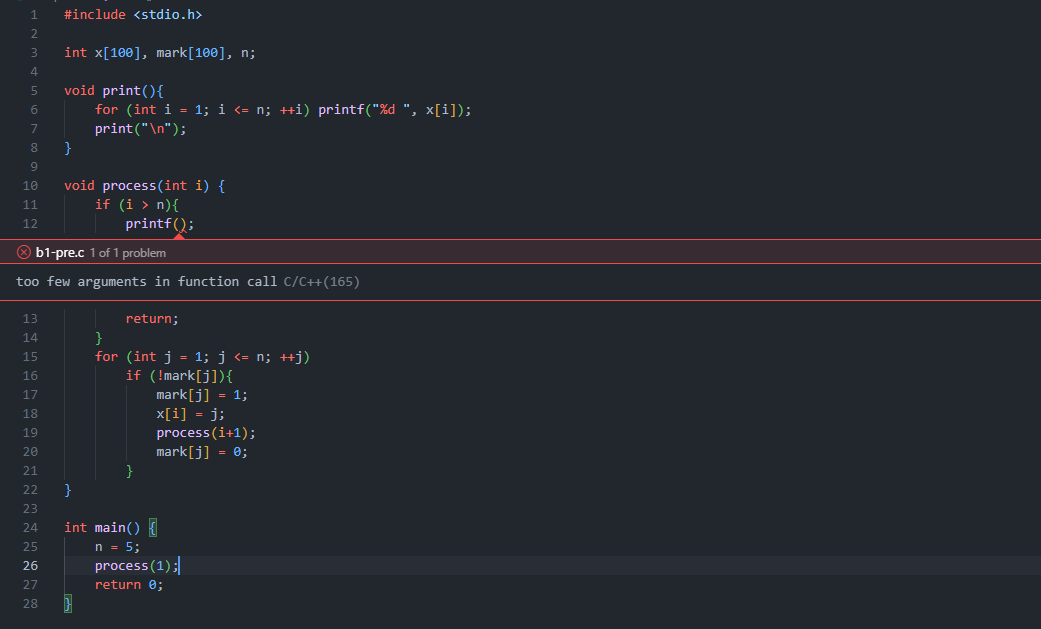
[Trước khi sửa 23](#_Toc140530285)

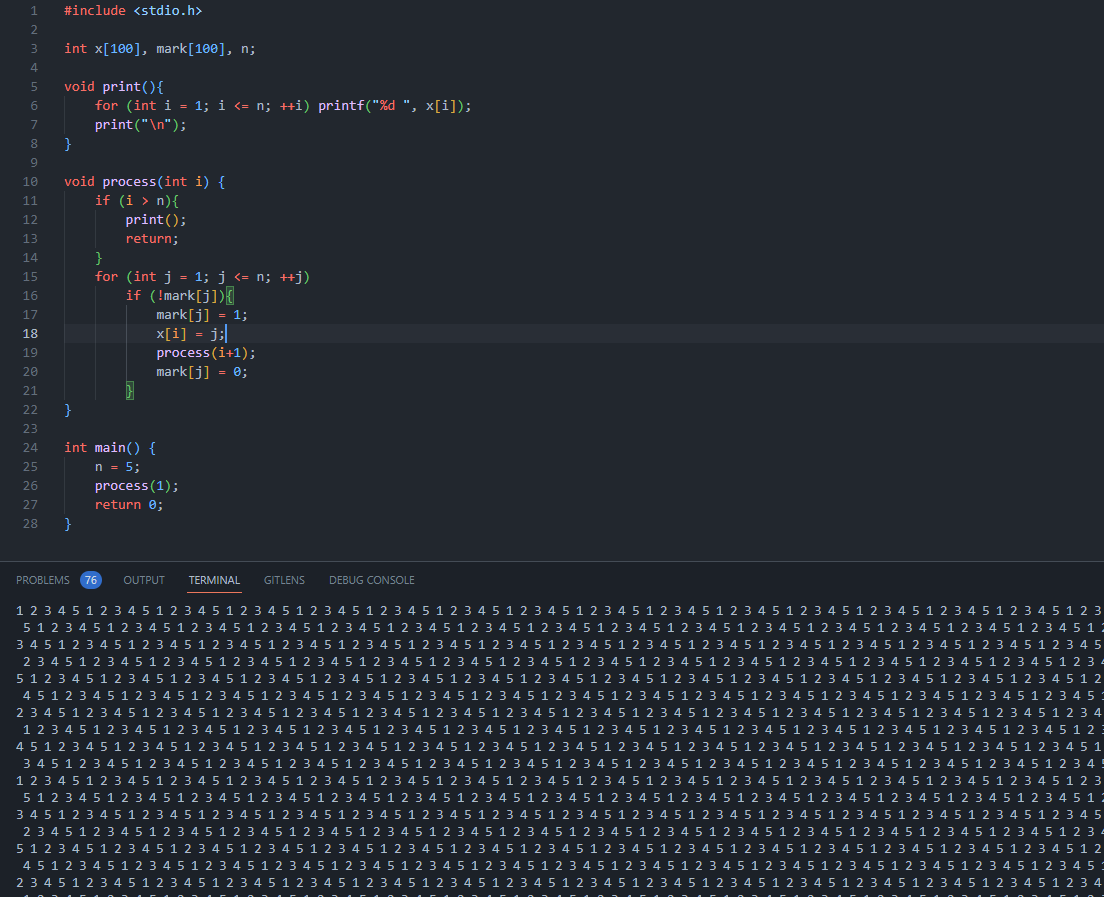
[Sau khi sửa 24](#_Toc140530286)

[Code 25](#_Toc140530287)

## Bài 1

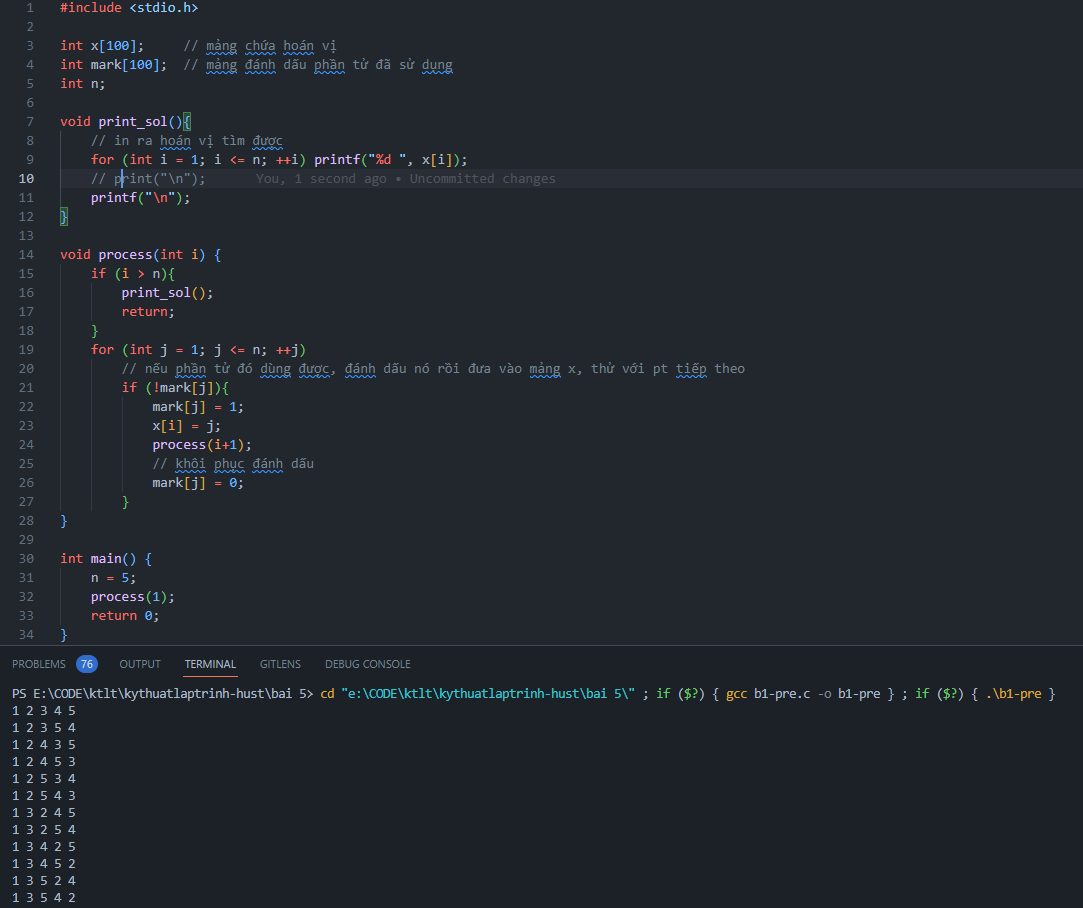
### Trước khi sửa

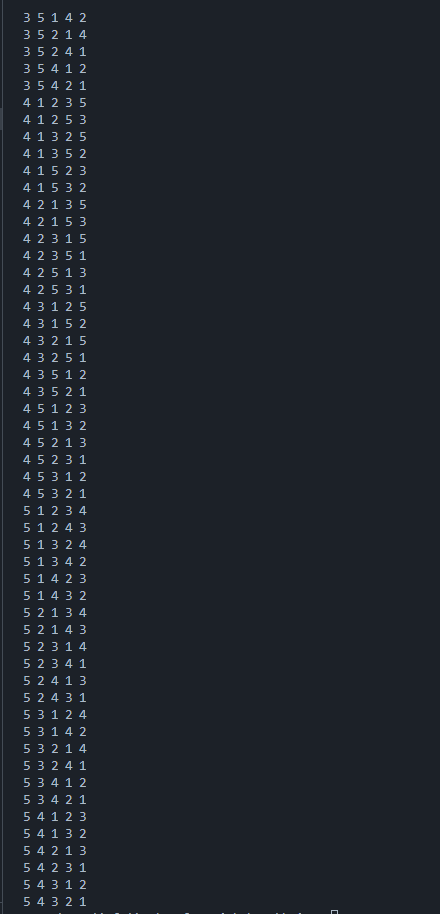




* Lỗi vòng lặp vô hạn

### Sau khi sửa





### Code

#include <stdio.h>

int x[100];     // mảng chứa hoán vị

int mark[100];  // mảng đánh dấu phần tử đã sử dụng

int n;

void print\_sol(){

    // in ra hoán vị tìm được

    for (int i = 1; i <= n; ++i) printf("%d ", x[i]);

    // print("\n");

    printf("\n");

}

void process(int i) {

    if (i > n){

        print\_sol();

        return;

    }

    for (int j = 1; j <= n; ++j)

        // nếu phần tử đó dùng được, đánh dấu nó rồi đưa vào mảng x, thử với pt tiếp theo

        if (!mark[j]){

            mark[j] = 1;

            x[i] = j;

            process(i+1);

            // khôi phục đánh dấu

            mark[j] = 0;

        }

}

int main() {

    n = 5;

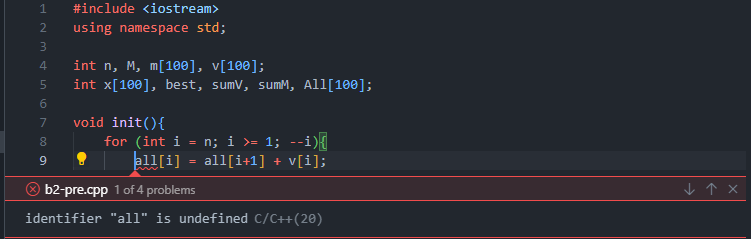
    process(1);

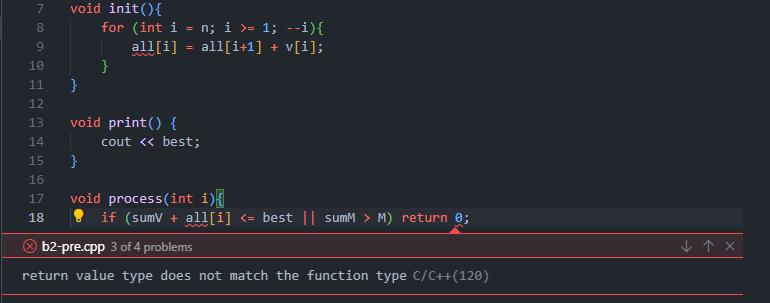
    return 0;

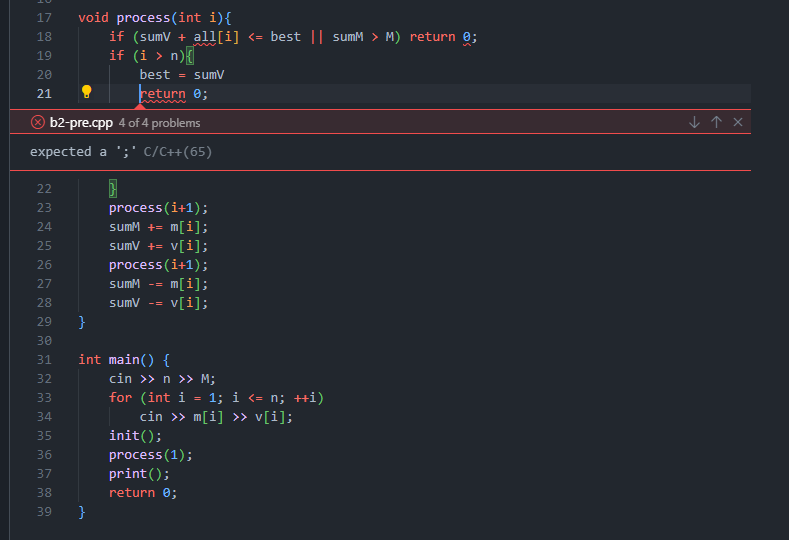
}

## Bài 2

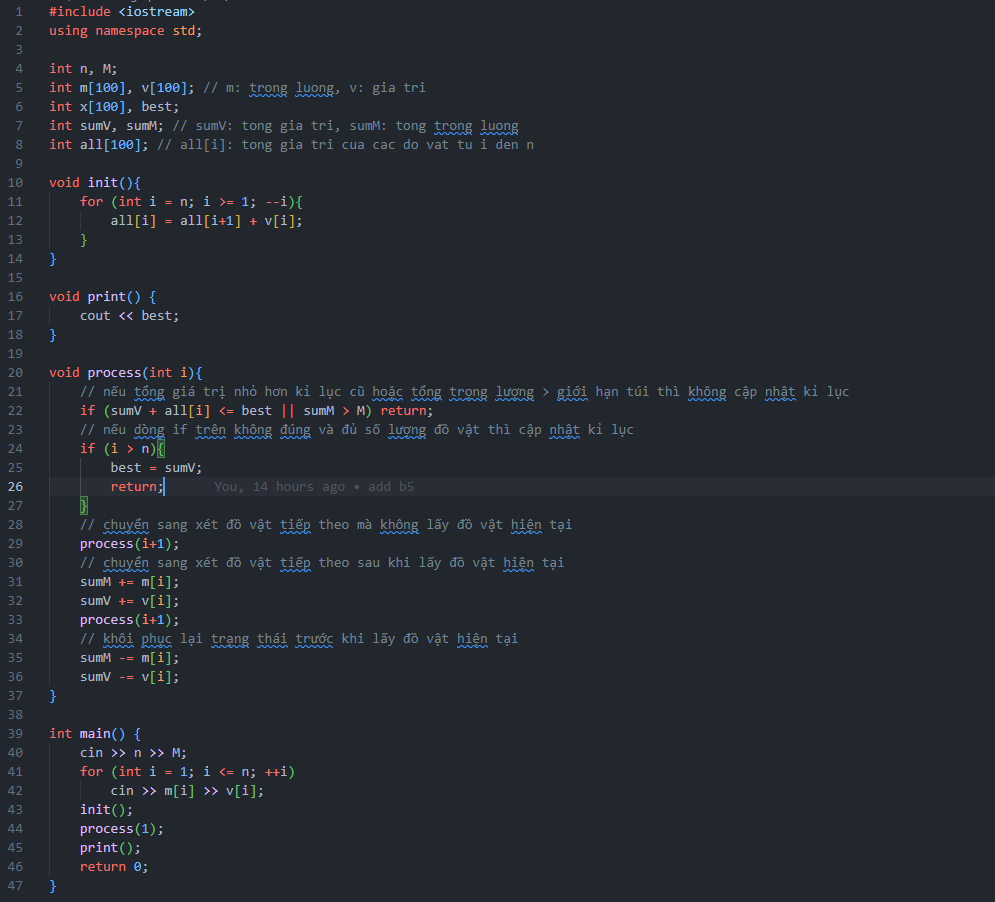
### Trước khi sửa

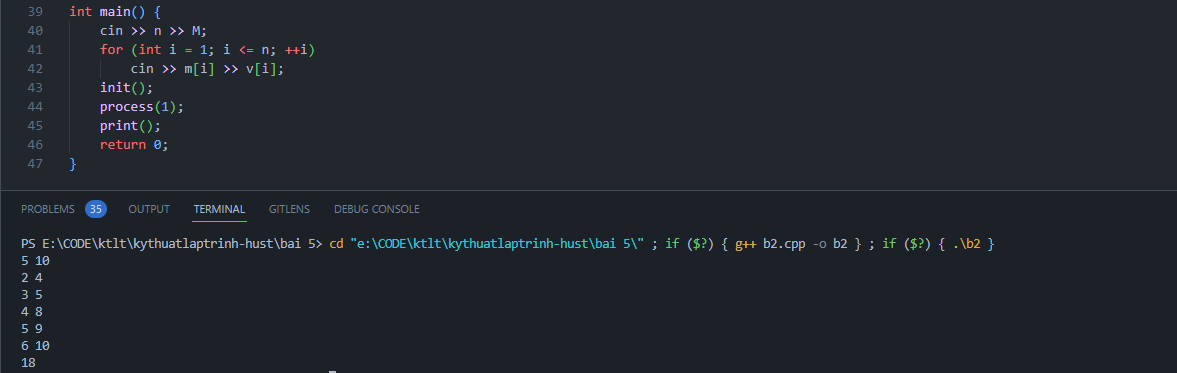






### Sau khi sửa





### Code

#include <iostream>

using namespace std;

int n, M;

int m[100], v[100]; // m: trong luong, v: gia tri

int x[100], best;

int sumV, sumM; // sumV: tong gia tri, sumM: tong trong luong

int all[100]; // all[i]: tong gia tri cua cac do vat tu i den n

void init(){

    for (int i = n; i >= 1; --i){

        all[i] = all[i+1] + v[i];

    }

}

void print() {

    cout << best;

}

void process(int i){

    // nếu tổng giá trị nhỏ hơn kỉ lục cũ hoặc tổng trọng lượng > giới hạn túi thì không cập nhật kỉ lục

    if (sumV + all[i] <= best || sumM > M) return;

    // nếu dòng if trên không đúng và đủ số lượng đồ vật thì cập nhật kỉ lục

    if (i > n){

        best = sumV;

        return;

    }

    // chuyển sang xét đồ vật tiếp theo mà không lấy đồ vật hiện tại

    process(i+1);

    // chuyển sang xét đồ vật tiếp theo sau khi lấy đồ vật hiện tại

    sumM += m[i];

    sumV += v[i];

    process(i+1);

    // khôi phục lại trạng thái trước khi lấy đồ vật hiện tại

    sumM -= m[i];

    sumV -= v[i];

}

int main() {

    cin >> n >> M;

    for (int i = 1; i <= n; ++i)

        cin >> m[i] >> v[i];

    init();

    process(1);

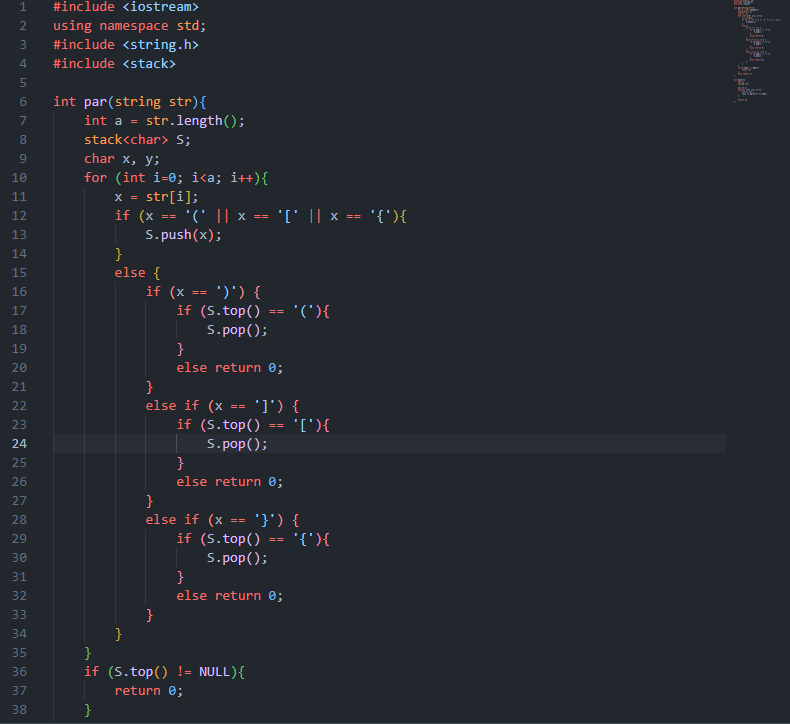
    print();

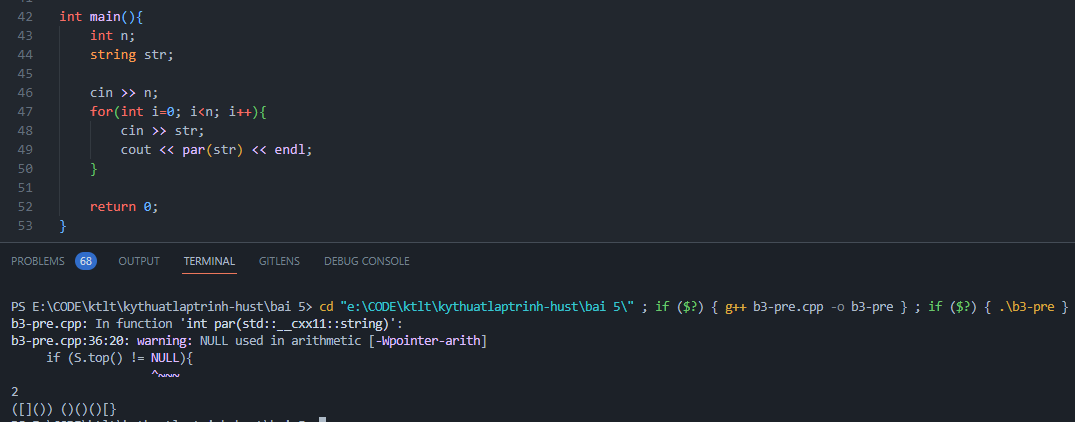
    return 0;

}

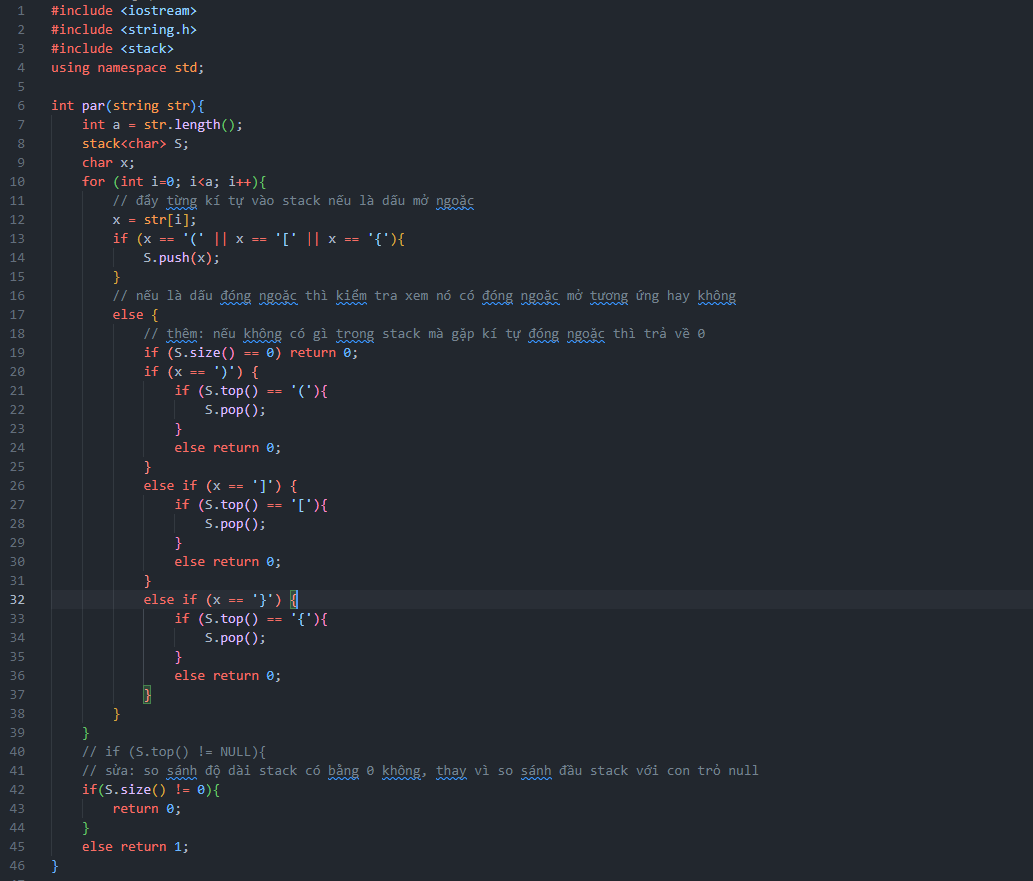
## Bài 3

### Trước khi sửa





### Sau khi sửa





### Code

#include <iostream>

#include <string.h>

#include <stack>

using namespace std;

int par(string str){

    int a = str.length();

    stack<char> S;

    char x;

    for (int i=0; i<a; i++){

        // đẩy từng kí tự vào stack nếu là dấu mở ngoặc

        x = str[i];

        if (x == '(' || x == '[' || x == '{'){

            S.push(x);

        }

        // nếu là dấu đóng ngoặc thì kiểm tra xem nó có đóng ngoặc mở tương ứng hay không

        else {

            // thêm: nếu không có gì trong stack mà gặp kí tự đóng ngoặc thì trả về 0

            if (S.size() == 0) return 0;

            if (x == ')') {

                if (S.top() == '('){

                    S.pop();

                }

                else return 0;

            }

            else if (x == ']') {

                if (S.top() == '['){

                    S.pop();

                }

                else return 0;

            }

            else if (x == '}') {

                if (S.top() == '{'){

                    S.pop();

                }

                else return 0;

            }

        }

    }

    // if (S.top() != NULL){

    // sửa: so sánh độ dài stack có bằng 0 không, thay vì so sánh đầu stack với con trỏ null

    if(S.size() != 0){

        return 0;

    }

    else return 1;

}

int main(){

    int n;

    string str;

    cin >> n;

    for(int i=0; i<n; i++){

        cin >> str;

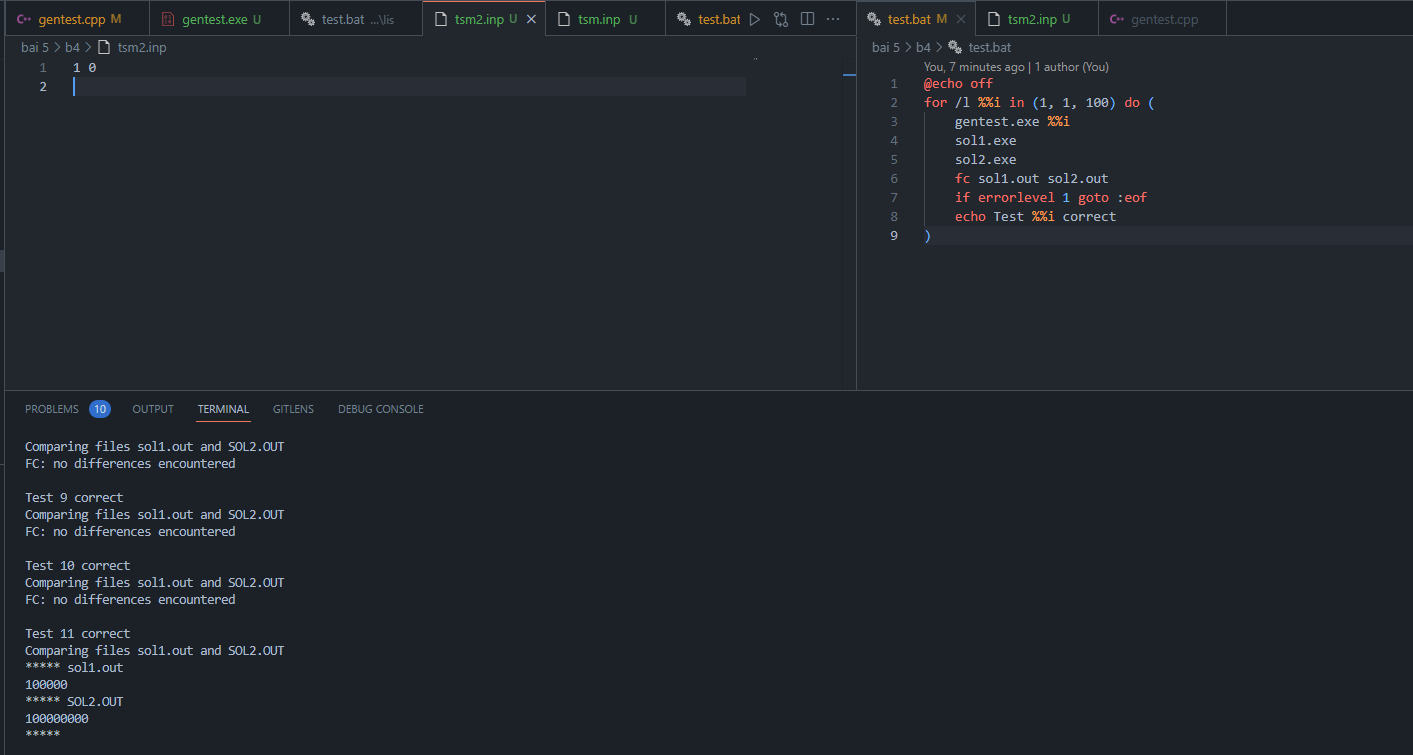
        cout << par(str) << endl;

    }

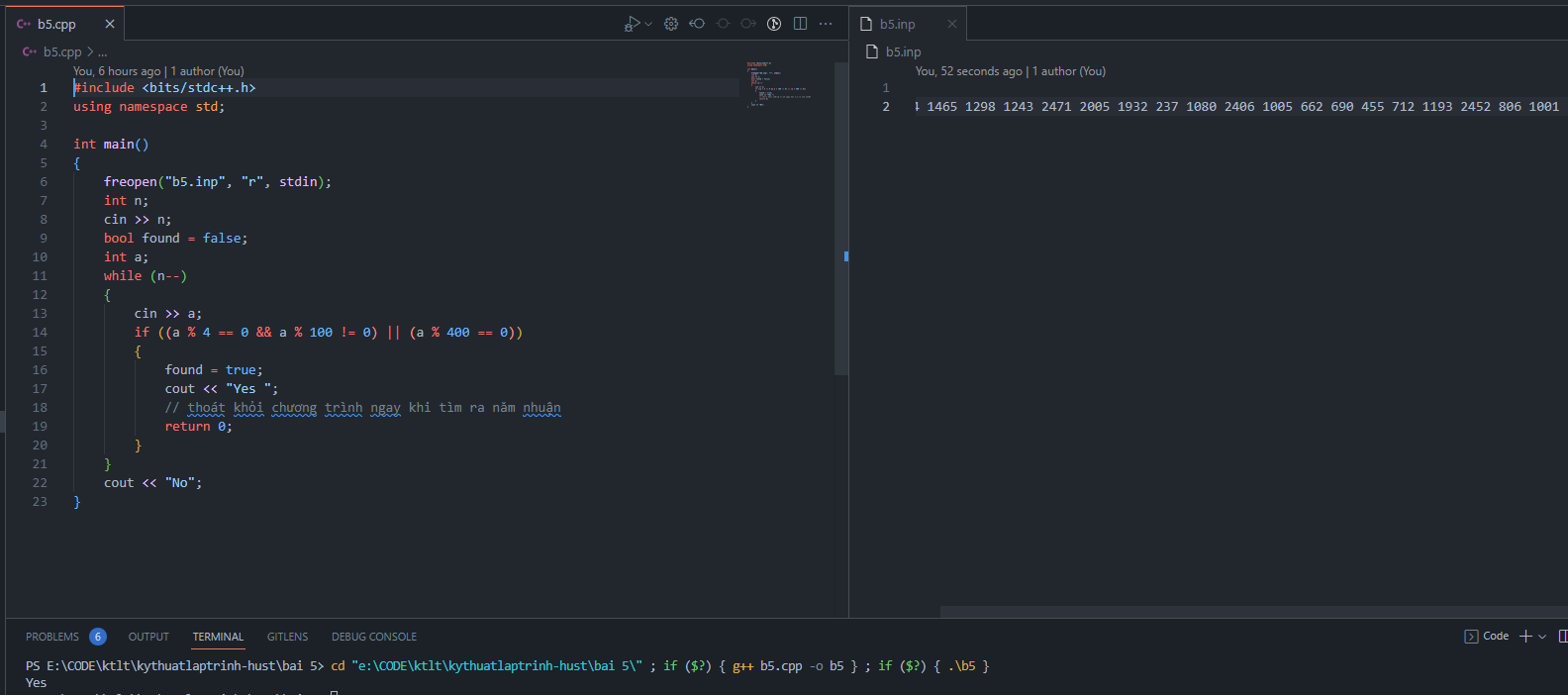
    return 0;

}

## Bài 4



## Bài 5



### Code

#include <bits/stdc++.h>

using namespace std;

int main()

{

    freopen("b5.inp", "r", stdin);

    int n;

    cin >> n;

    bool found = false;

    int a;

    while (n--)

    {

        cin >> a;

        if ((a % 4 == 0 && a % 100 != 0) || (a % 400 == 0))

        {

            found = true;

            cout << "Yes ";

            // thoát khỏi chương trình ngay khi tìm ra năm nhuận

            return 0;

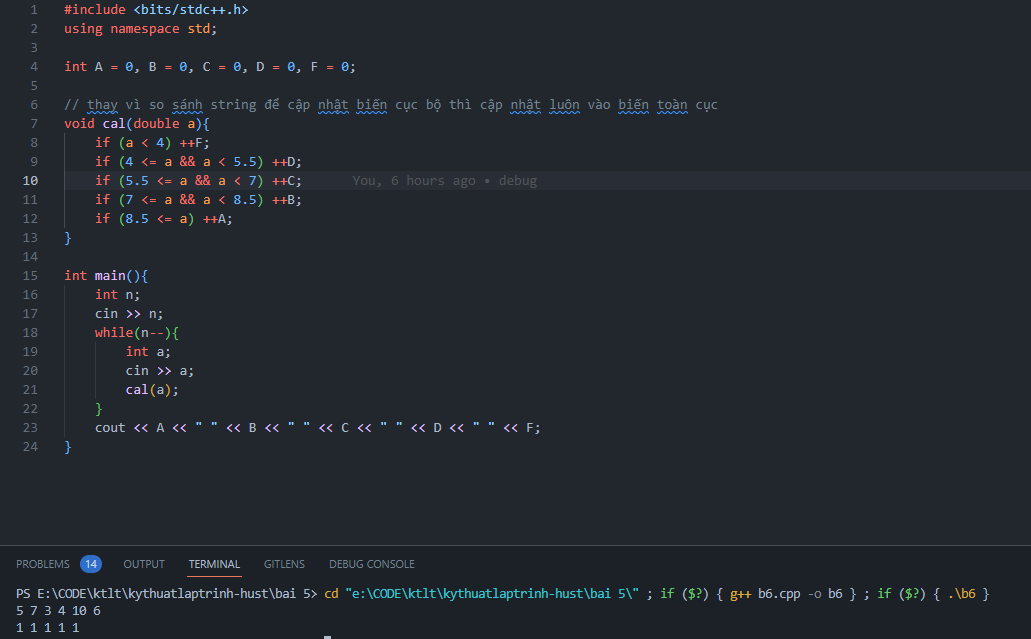
        }

    }

    cout << "No";

}

## Bài 6



### Code

#include <bits/stdc++.h>

using namespace std;

int A = 0, B = 0, C = 0, D = 0, F = 0;

// thay vì so sánh string để cập nhật biến cục bộ thì cập nhật luôn vào biến toàn cục

void cal(double a){

    if (a < 4) ++F;

    if (4 <= a && a < 5.5) ++D;

    if (5.5 <= a && a < 7) ++C;

    if (7 <= a && a < 8.5) ++B;

    if (8.5 <= a) ++A;

}

int main(){

    int n;

    cin >> n;

    while(n--){

        int a;

        cin >> a;

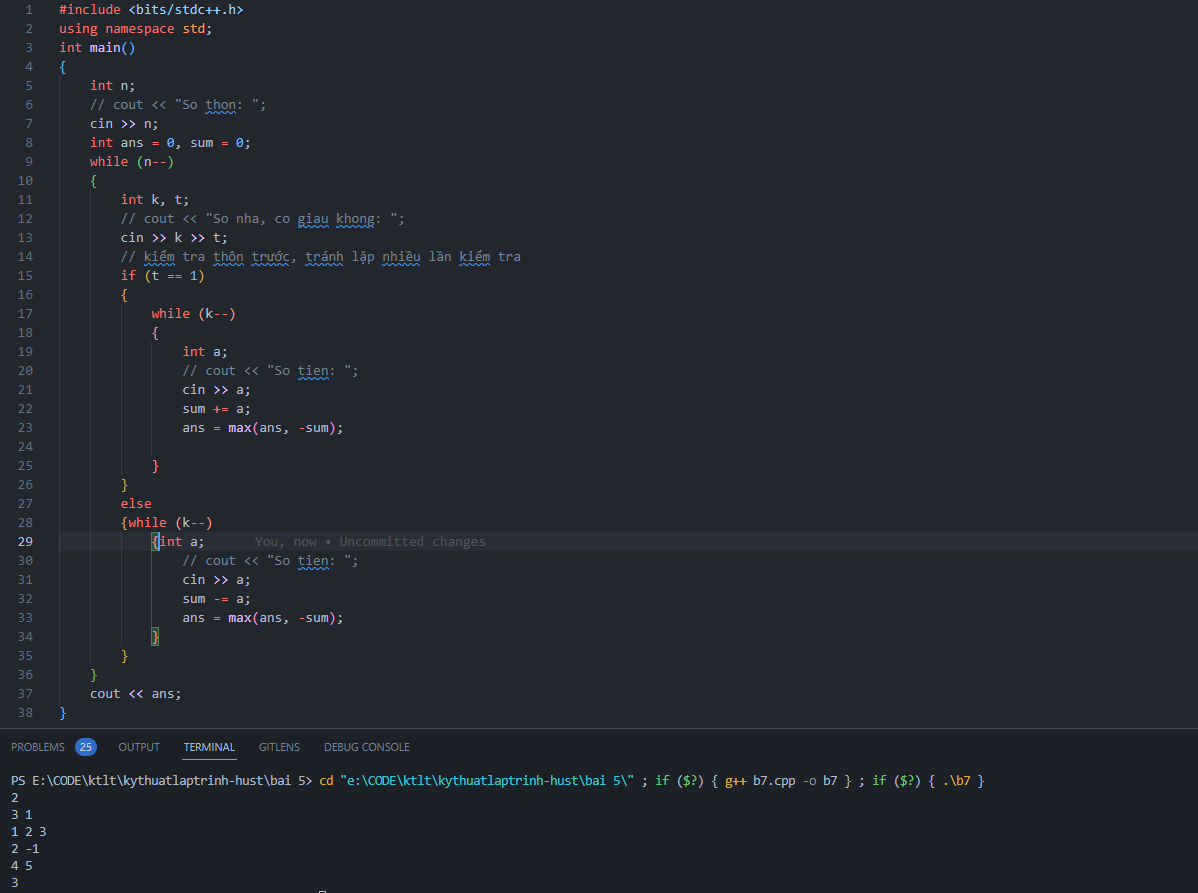
        cal(a);

    }

    cout << A << " " << B << " " << C << " " << D << " " << F;

}

## Bài 7



### Code

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int n;

    // cout << "So thon: ";

    cin >> n;

    int ans = 0, sum = 0;

    while (n--)

    {

        int k, t;

        // cout << "So nha, co giau khong: ";

        cin >> k >> t;

        // kiểm tra thôn trước, tránh lặp nhiều lần kiểm tra

        if (t == 1)

        {

            while (k--)

            {

                int a;

                // cout << "So tien: ";

                cin >> a;

                sum += a;

                ans = max(ans, -sum);

            }

        }

        else

        {while (k--)

            {int a;

                // cout << "So tien: ";

                cin >> a;

                sum -= a;

                ans = max(ans, -sum);

            }

        }

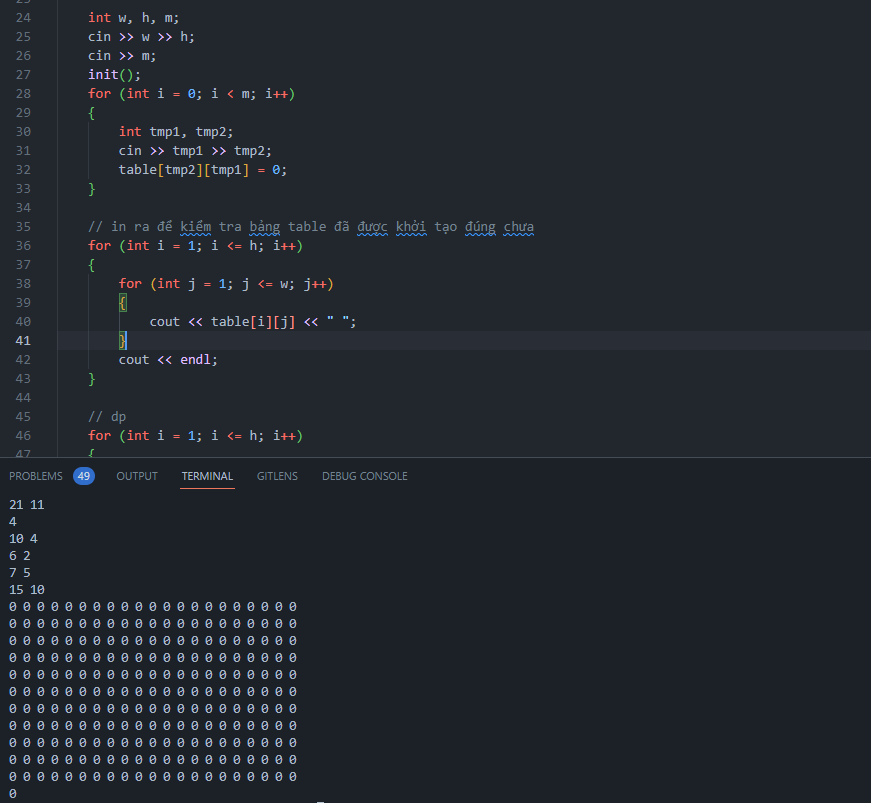
    }

    cout << ans;

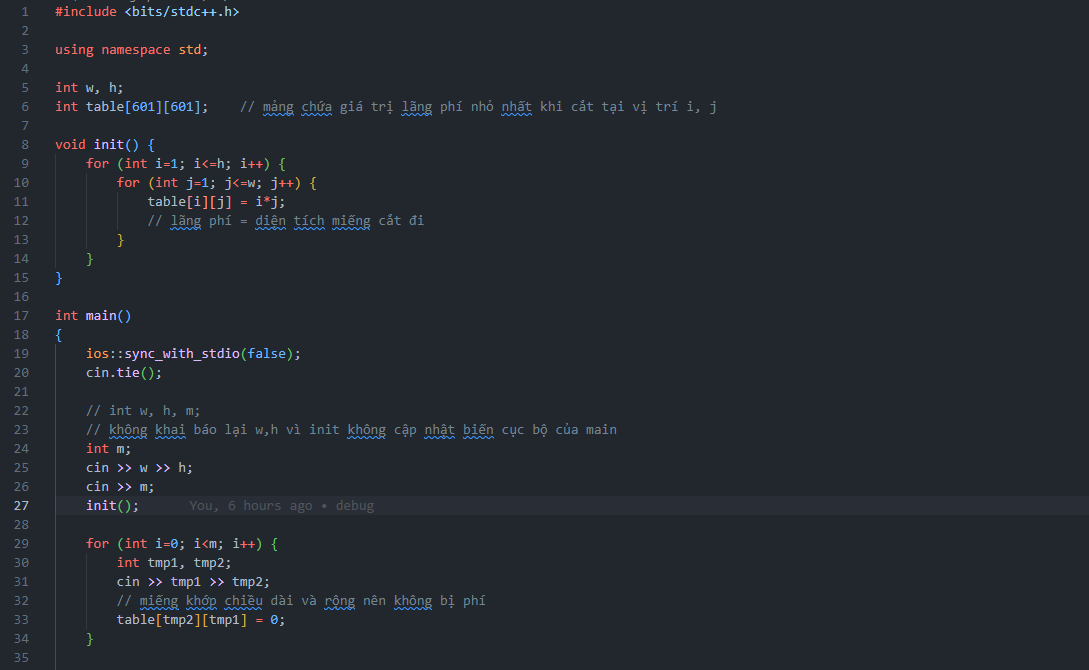
}

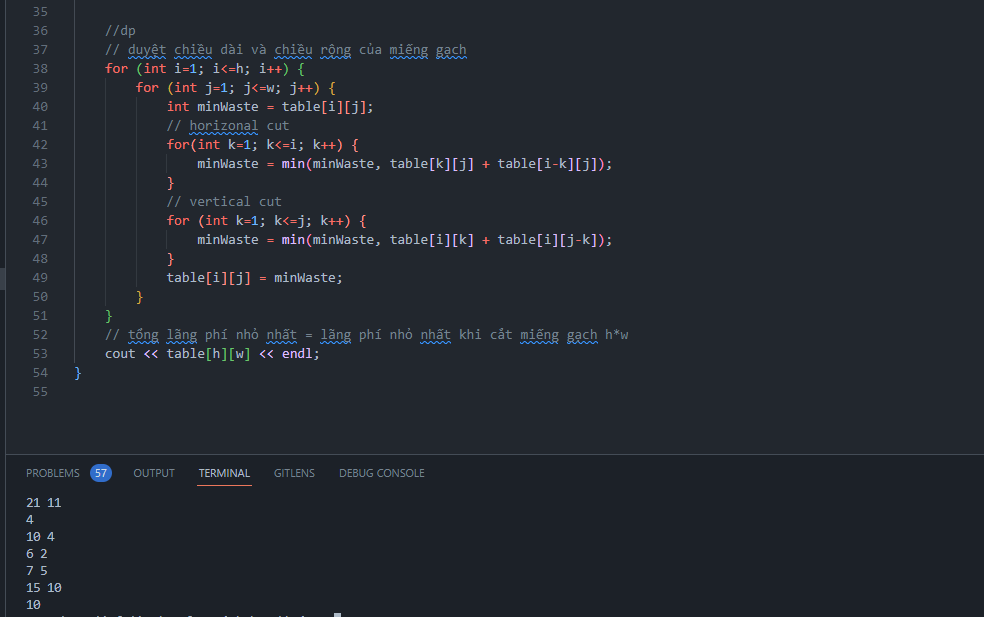
## Bài 8

### Trước khi sửa



### Sau khi sửa





### Code

#include <bits/stdc++.h>

using namespace std;

int w, h;

int table[601][601];    // mảng chứa giá trị lãng phí nhỏ nhất khi cắt tại vị trí i, j

void init() {

    for (int i=1; i<=h; i++) {

        for (int j=1; j<=w; j++) {

            table[i][j] = i\*j;

            // lãng phí = diện tích miếng cắt đi

        }

    }

}

int main()

{

    ios::sync\_with\_stdio(false);

    cin.tie();

    // int w, h, m;

    // không khai báo lại w,h vì init không cập nhật biến cục bộ của main

    int m;

    cin >> w >> h;

    cin >> m;

    init();

    for (int i=0; i<m; i++) {

        int tmp1, tmp2;

        cin >> tmp1 >> tmp2;

        // miếng khớp chiều dài và rộng nên không bị phí

        table[tmp2][tmp1] = 0;

    }

    //dp

    // duyệt chiều dài và chiều rộng của miếng gạch

    for (int i=1; i<=h; i++) {

        for (int j=1; j<=w; j++) {

            int minWaste = table[i][j];

            // horizonal cut

            for(int k=1; k<=i; k++) {

                minWaste = min(minWaste, table[k][j] + table[i-k][j]);

            }

            // vertical cut

            for (int k=1; k<=j; k++) {

                minWaste = min(minWaste, table[i][k] + table[i][j-k]);

            }

            table[i][j] = minWaste;

        }

    }

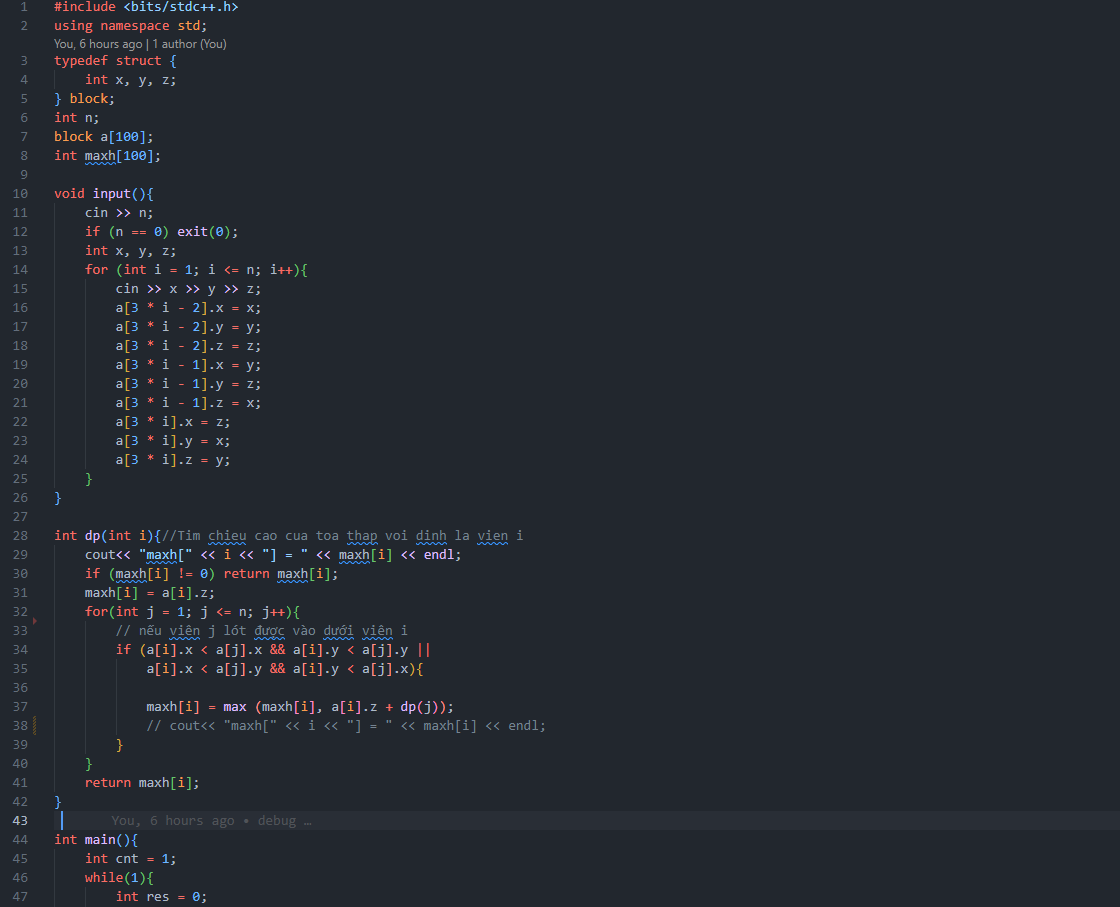
    // tổng lãng phí nhỏ nhất = lãng phí nhỏ nhất khi cắt miếng gạch h\*w

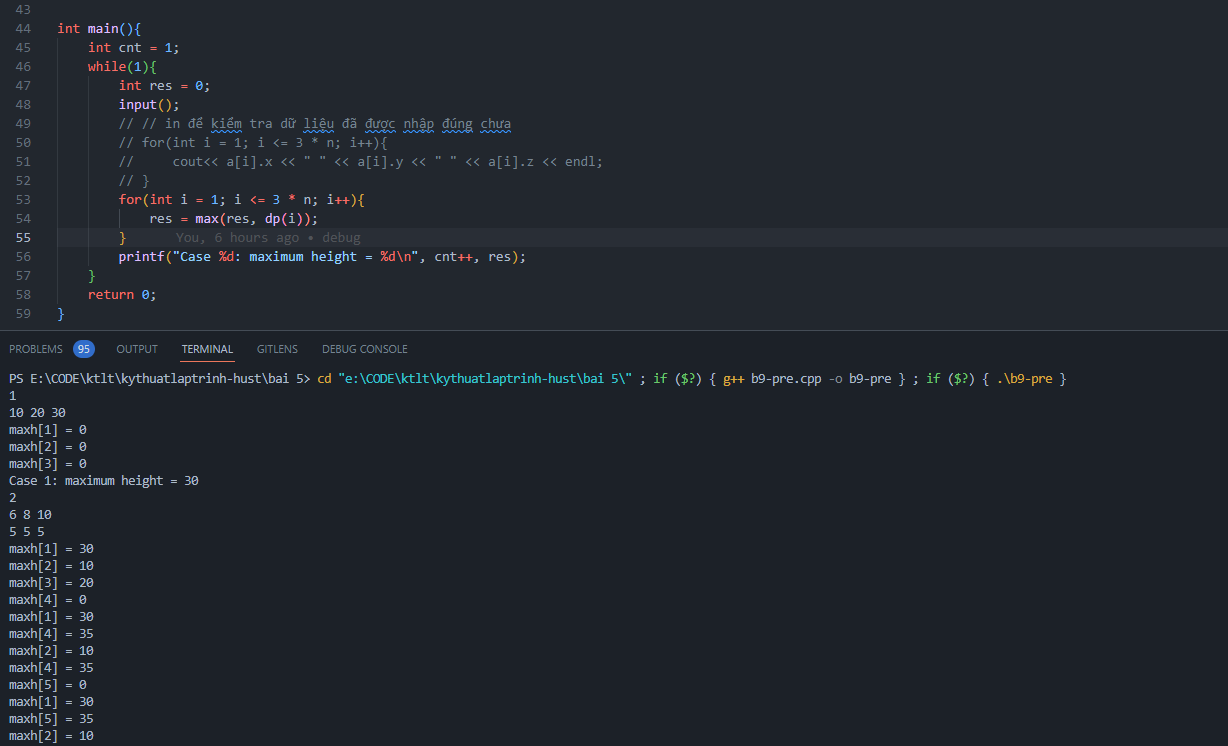
    cout << table[h][w] << endl;

}

## Bài 9

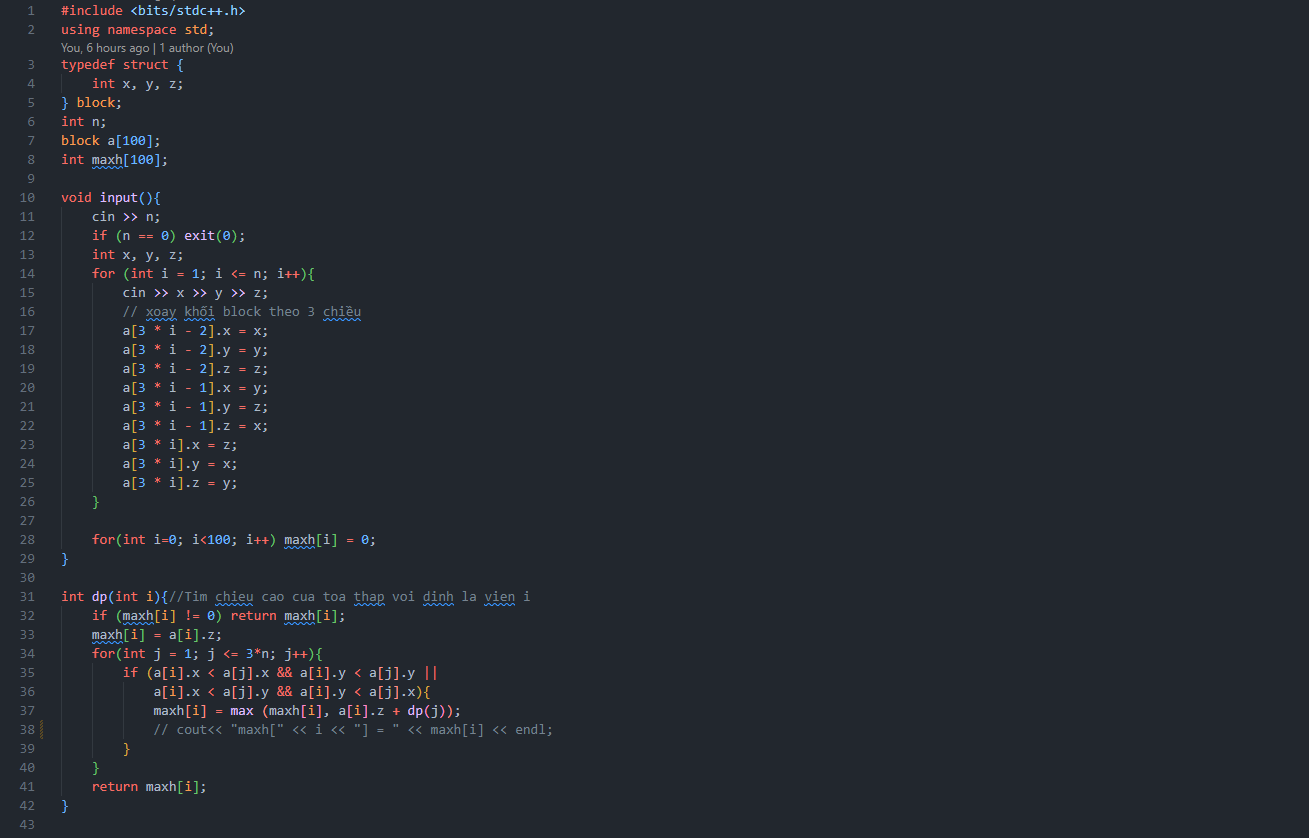
### Trước khi sửa

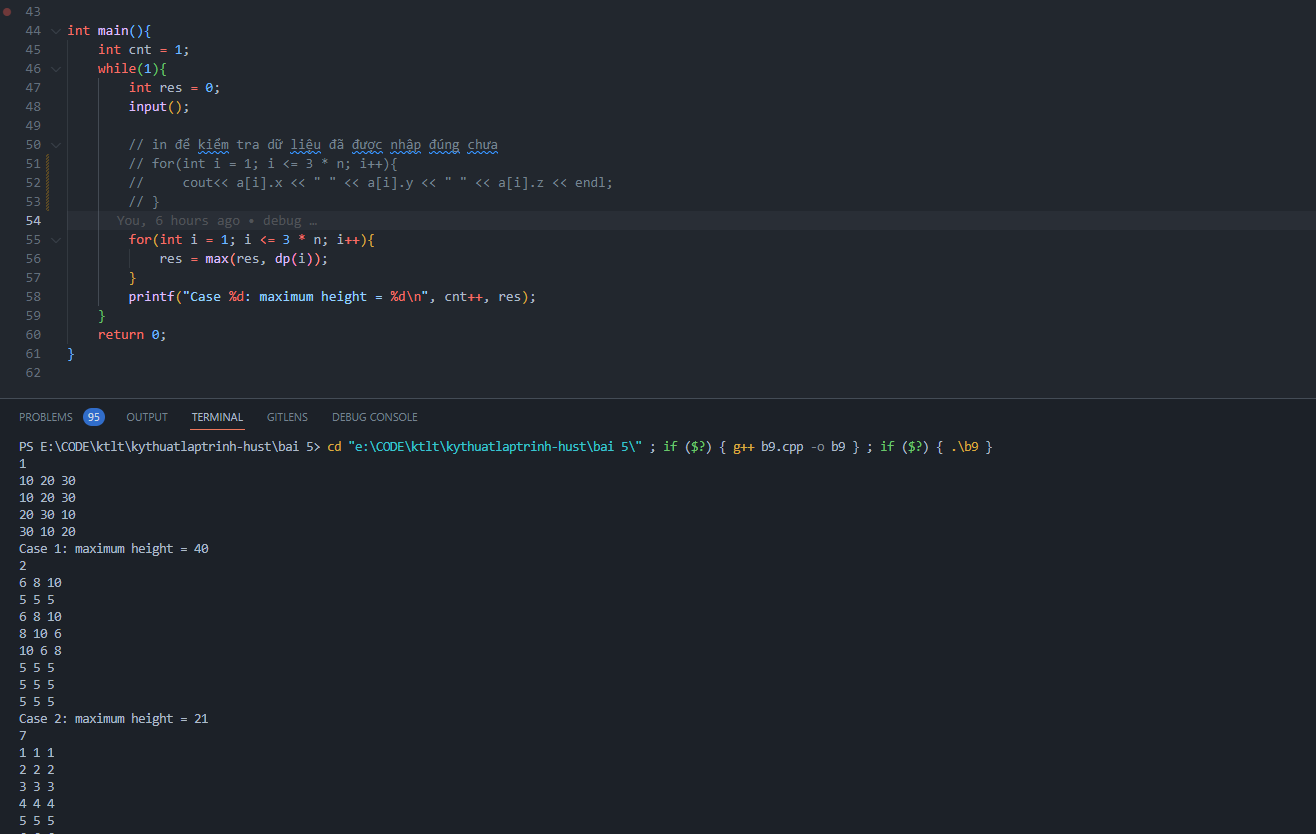


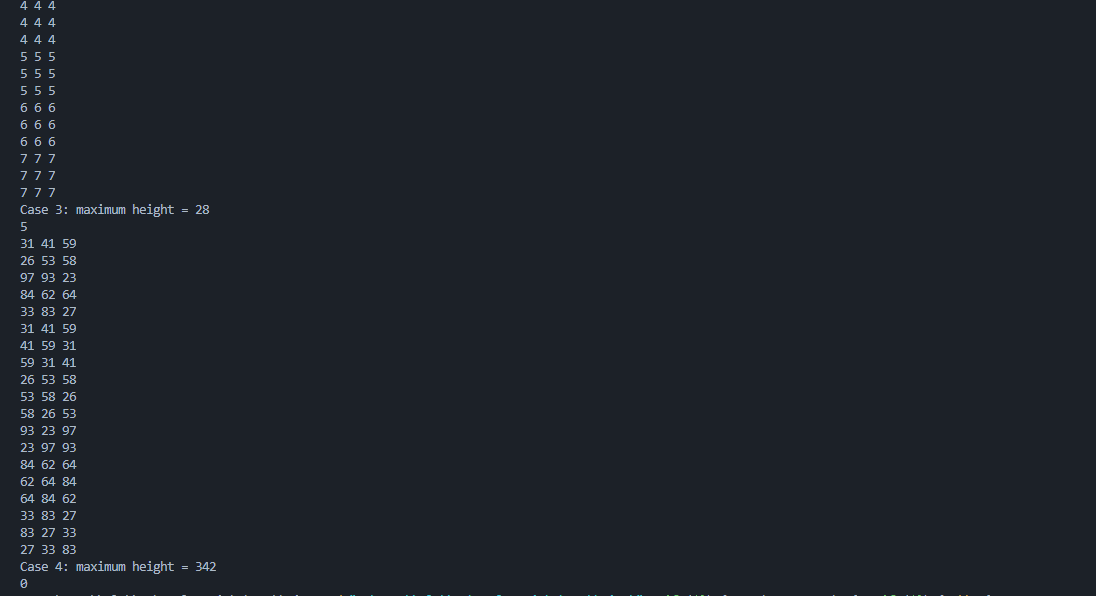


* Lỗi do max[i] chưa được khởi tạo đúng

### Sau khi sửa







### Code

#include <bits/stdc++.h>

using namespace std;

typedef struct {

    int x, y, z;

} block;

int n;

block a[100];

int maxh[100];

void input(){

    cin >> n;

    if (n == 0) exit(0);

    int x, y, z;

    for (int i = 1; i <= n; i++){

        cin >> x >> y >> z;

        // xoay khối block theo 3 chiều

        a[3 \* i - 2].x = x;

        a[3 \* i - 2].y = y;

        a[3 \* i - 2].z = z;

        a[3 \* i - 1].x = y;

        a[3 \* i - 1].y = z;

        a[3 \* i - 1].z = x;

        a[3 \* i].x = z;

        a[3 \* i].y = x;

        a[3 \* i].z = y;

    }

    for(int i=0; i<100; i++) maxh[i] = 0;

}

int dp(int i){//Tim chieu cao cua toa thap voi dinh la vien i

    if (maxh[i] != 0) return maxh[i];

    maxh[i] = a[i].z;

    for(int j = 1; j <= 3\*n; j++){

        if (a[i].x < a[j].x && a[i].y < a[j].y ||

            a[i].x < a[j].y && a[i].y < a[j].x){

            maxh[i] = max (maxh[i], a[i].z + dp(j));

            // cout<< "maxh[" << i << "] = " << maxh[i] << endl;

        }

    }

    return maxh[i];

}

int main(){

    int cnt = 1;

    while(1){

        int res = 0;

        input();

        // in để kiểm tra dữ liệu đã được nhập đúng chưa

        // for(int i = 1; i <= 3 \* n; i++){

        //     cout<< a[i].x << " " << a[i].y << " " << a[i].z << endl;

        // }

        for(int i = 1; i <= 3 \* n; i++){

            res = max(res, dp(i));

        }

        printf("Case %d: maximum height = %d\n", cnt++, res);

    }

    return 0;

}