Contents

[Bài thực hành số 3 6](#_Toc154342538)

[**Bài 5.1:** Tìm và sửa các lỗi cú pháp Đoạn code sau liệt kê tất cả các hoán vị n số. Hãy tìm và sửa các lỗi cú pháp như hướng dẫn ở trên. 6](#_Toc154342539)

[**Bài 5.2:** Tìm và sửa các lỗi cú pháp Bài toán cái túi: Cho một cái túi có sức chứa M và n đồ vật. Đồ vật thứ i có khối lượng m i và giá trị v i . Cần chọn ra một số đồ vật để bỏ vào túi sao cho tổng khối lượng không quá M và tổng giá trị là lớn nhất có thể. Đoạn code sau đây giải bài toán cái túi bằng phương pháp duyệt nhánh cận. Hãy tìm và sửa các lỗi cú pháp 9](#_Toc154342540)

[**Bài 5.3:** Dãy ngoặc đúng Viết một chương trình nhận vào một dãy dấu ngoặc và kiểu tra xem dãy dấu ngoặc đóng mở đúng chưa. Đề bài: http://codeforces.com/group/Ir5CI6f3FD/contest/269186/problem/H Mã nguồn dưới đây là của một sinh viên, khi submit bị lỗi runtime (Exit code is -1073741819). Sử dụng công cụ debug ở trên, hãy tìm và sửa các lỗi trong mã nguồn 13](#_Toc154342541)

[**Bài 5.4:** Bài toán người du lịch Một người xuất phát tại thành phố 1, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng 1 lần và quay về 1. Chi phí để đi từ thành phố 𝑖 sang thành phố 𝑗 là ci,j. Hãy tìm tổng chi phí nhỏ nhất có thể 18](#_Toc154342542)

[**Bài 5.5:** Năm nhuận Một năm được coi là nhuận nếu hoặc nó chia hết cho 4 nhưng không chia hết cho 100, hoặc nó chia hết cho 400. Cho một danh sách các năm, kiểm tra xem có tồn tại năm nhuận trong danh sách đó hay không. Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình. 23](#_Toc154342543)

[**Bài 5.6:** Tổng kết Một lớp có n sinh viên. Sinh viên thứ i có điểm tổng kết là ai theo thang điểm 10. Để đánh giá chất lượng dạy học, giảng viên muốn biết có bao nhiêu bạn đạt điểm A, B, C, D, F. Quy đổi thang điểm được cho như sau: a<4: F 4≤a<5.5: D 5.5≤a<7: C 7≤a<8.5: B 8.5≤a: A Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình. 26](#_Toc154342544)

[**Bài 5.7**. Chia tiền Sau đại dịch, thầy trò Đường Tăng muốn xin tiền của các nhà giàu để chia cho các nhà nghèo. Họ sẽ vào n thôn, thôn thứ i có ki nhà. Mỗi thôn họ sẽ quyết định xin tiền hay cho tiền, phụ thuộc vào đánh giá của họ về mức độ giàu nghèo ở đây. Nếu thôn i giàu, họ sẽ đi từng nhà trong số ki nhà này và xin ai,j tiền của nhà thứ j. Nếu thôn i nghèo, họ sẽ đi từng nhà trong số ki nhà này và phát ai,j tiền cho nhà thứ j. Hãy tính số tiền ít nhất họ phải mang theo để đảm bảo có thể phát đủ cho người nghèo (tức số tiền luôn không bị âm) Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình. 30](#_Toc154342545)

[**Bài 5.8:** Cắt hình chữ nhật Phong là một nhà điêu khắc, ông có một tấm đá cẩm thạch hình chữ nhật kích thước W×H . Ông ta muốn cắt tấm đá thành các miếng hình chữ nhật kích thước W1×H1,W2×H2,…,WN×HN . Ông ta muốn cắt đến tối đa các mẫu kích thước có thể. Tấm đá có những vân đá cho nên không thể xoay khi sử dụng, có nghĩa là không thể cắt ra miếng B×A thay cho miếng A×B trừ khi A=B . Các miếng phải được cắt tại các điểm nguyên trên hàng cột và mỗi nhát cắt phải cắt đến hết hàng hoặc hết cột. Sau khi cắt sẽ còn lại những mẩu đá còn thừa bỏ đi, nghĩa là những mẩu đá không thể cắt thành miếng kích thước cho trước nào. Yêu cầu: Hãy tìm cách cắt sao cho còn ít nhất diện tích đá thừa bỏ đi. Sử dụng công cụ debug ở trên, hãy tìm và sửa các lỗi trong mã nguồn dưới đây 33](#_Toc154342546)

[**Bài 5.9:** Xây tháp Perhaps you have heard of the legend of the Tower of Babylon. Nowadays many details of this tale have been forgotten. So now, in line with the educational nature of this contest, we will tell you the whole story: The babylonians had n types of blocks, and an unlimited supply of blocks of each type. Each type-i block was a rectangular solid with linear dimensions (xi,yi,zi). A block could be reoriented so that any two of its three dimensions determined the dimensions of the base and the other dimension was the height. They wanted to construct the tallest tower possible by stacking blocks. The problem was that, in building a tower, one block could only be placed on top of another block as long as the two base dimensions of the upper block were both strictly smaller than the corresponding base dimensions of the lower block. This meant, for example, that blocks oriented to have equal-sized bases couldn't be stacked. Your job is to write a program that determines the height of the tallest tower the babylonians can build with a given set of blocks. 38](#_Toc154342547)

[Hình 1. Code bài 5.1 ảnh 1 4](#_Toc154342609)

[Hình 2. Code bài 5.1 ảnh 2 4](#_Toc154342610)

[Hình 3. Output bài 5.1 ảnh 1 5](#_Toc154342611)

[Hình 4. Output bài 5.1 ảnh 2 5](#_Toc154342612)

[Hình 5. Code bài 5.2 ảnh 1 7](#_Toc154342613)

[Hình 6. Code bài 5.2 ảnh 2 8](#_Toc154342614)

[Hình 7. Code bài 5.2 ảnh 3 8](#_Toc154342615)

[Hình 8. Output bài 5.2 9](#_Toc154342616)

[Hình 9. Code bài 5.3 ảnh 1 12](#_Toc154342617)

[Hình 10. Code bài 5.3 ảnh 2 12](#_Toc154342618)

[Hình 11. Code bài 5.3 ảnh 3 13](#_Toc154342619)

[Hình 12. Output bài 5.3 13](#_Toc154342620)

[Hình 13. Code bài 5.4 ảnh 1 16](#_Toc154342621)

[Hình 14. Code bài 5.4 ảnh 2 16](#_Toc154342622)

[Hình 15. Code bài 5.4 ảnh 3 17](#_Toc154342623)

[Hình 16. Code bài 5.4 ảnh 4 17](#_Toc154342624)

[Hình 17. Output bài 5.4 18](#_Toc154342625)

[Hình 18. Code bài 5.5 ảnh 1 21](#_Toc154342626)

[Hình 19. Code bài 5.5 ảnh 2 21](#_Toc154342627)

[Hình 20. Output bài 5.5 22](#_Toc154342628)

[Hình 22. Code bào 5.6 ảnh 1 24](#_Toc154342629)

[Hình 23. Code bào 5.6 ảnh 2 25](#_Toc154342630)

[Hình 24. Output bài 5.6 25](#_Toc154342631)

[Hình 26. Code bài 5.7 ảnh 1 28](#_Toc154342632)

[Hình 27. Code bài 5.7 ảnh 2 28](#_Toc154342633)

[Hình 31. Output bài 5.7 29](#_Toc154342634)

[Hình 32. Code bài 5.8 ảnh 1 31](#_Toc154342635)

[Hình 33. Code bài 5.8 ảnh 2 32](#_Toc154342636)

[Hình 34. Code bài 5.8 ảnh 3 32](#_Toc154342637)

[Hình 37. Output bài 5,8 33](#_Toc154342638)

[Hình 39. Code bài 5.9 ảnh 1 36](#_Toc154342639)

[Hình 40. Code bài 5.9 ảnh 2 37](#_Toc154342640)

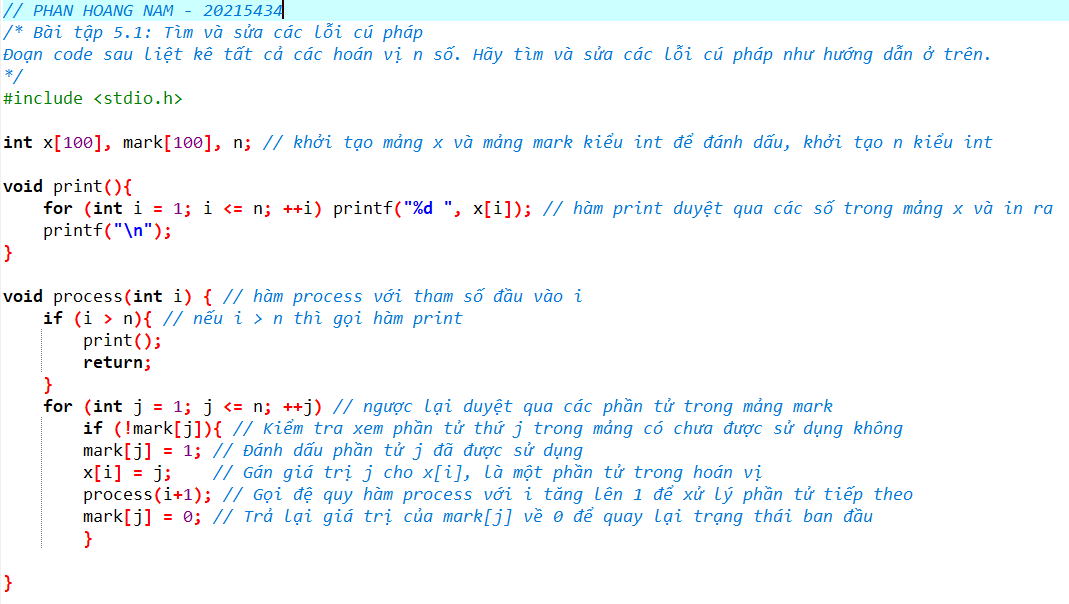
[Hình 41. Code bài 5.9 ảnh 3 37](#_Toc154342641)

[Hình 42. Code bài 5.9 ảnh 4 38](#_Toc154342642)

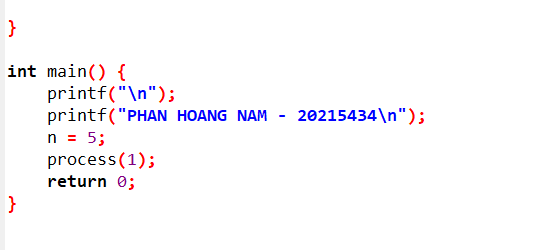
[Hình 43. Output bài 5.9 39](#_Toc154342643)

# Bài thực hành số 5

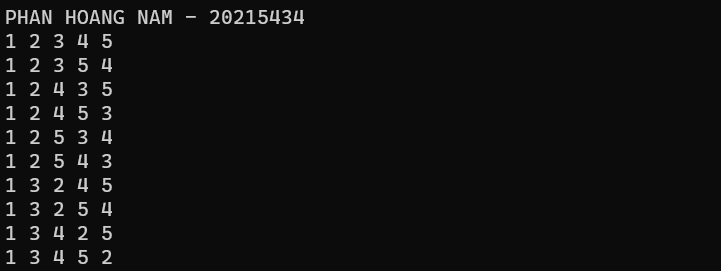
## **Bài 5.1:** Tìm và sửa các lỗi cú pháp Đoạn code sau liệt kê tất cả các hoán vị n số. Hãy tìm và sửa các lỗi cú pháp như hướng dẫn ở trên.



Hình 1. Code bài 5.1 ảnh 1



Hình 2. Code bài 5.1 ảnh 2



Hình 3. Output bài 5.1 ảnh 1



Hình 4. Output bài 5.1 ảnh 2

// PHAN HOANG NAM - 20215434

/\* Bài tập 5.1: Tìm và sửa các lỗi cú pháp

Đoạn code sau liệt kê tất cả các hoán vị n số. Hãy tìm và sửa các lỗi cú pháp như hướng dẫn ở trên.

\*/

#include <stdio.h>

int x[100], mark[100], n; // khởi tạo mảng x và mảng mark kiểu int để đánh dấu, khởi tạo n kiểu int

void print(){

for (int i = 1; i <= n; ++i) printf("%d ", x[i]); // hàm print duyệt qua các số trong mảng x và in ra

printf("\n");

}

void process(int i) { // hàm process với tham số đầu vào i

if (i > n){ // nếu i > n thì gọi hàm print

print();

return;

}

for (int j = 1; j <= n; ++j) // ngược lại duyệt qua các phần tử trong mảng mark

if (!mark[j]){ // Kiểm tra xem phần tử thứ j trong mảng có chưa được sử dụng không

mark[j] = 1; // Đánh dấu phần tử j đã được sử dụng

x[i] = j; // Gán giá trị j cho x[i], là một phần tử trong hoán vị

process(i+1); // Gọi đệ quy hàm process với i tăng lên 1 để xử lý phần tử tiếp theo

mark[j] = 0; // Trả lại giá trị của mark[j] về 0 để quay lại trạng thái ban đầu

}

}

int main() {

printf("\n");

printf("PHAN HOANG NAM - 20215434\n");

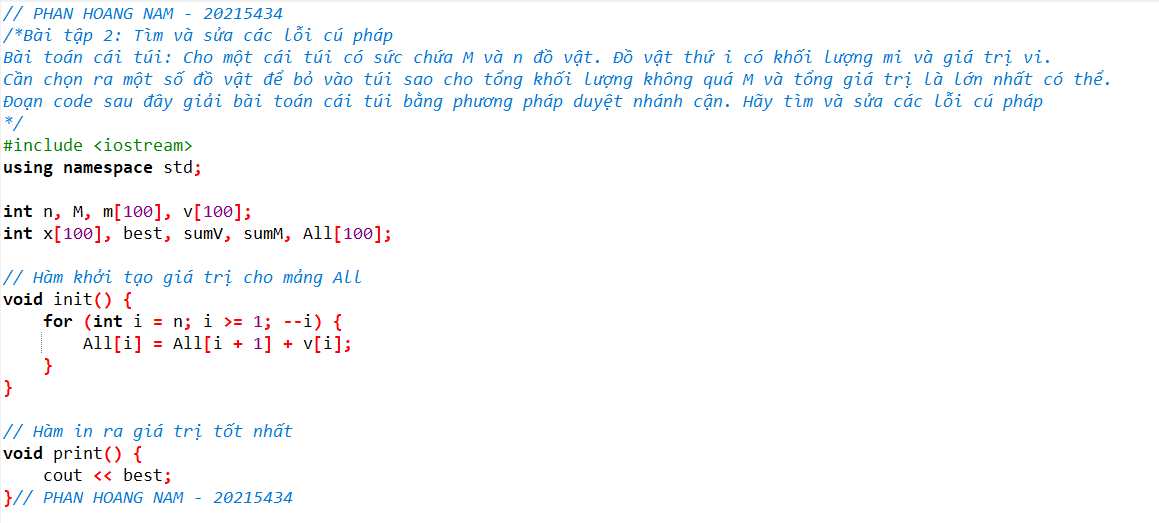
n = 5;

process(1);

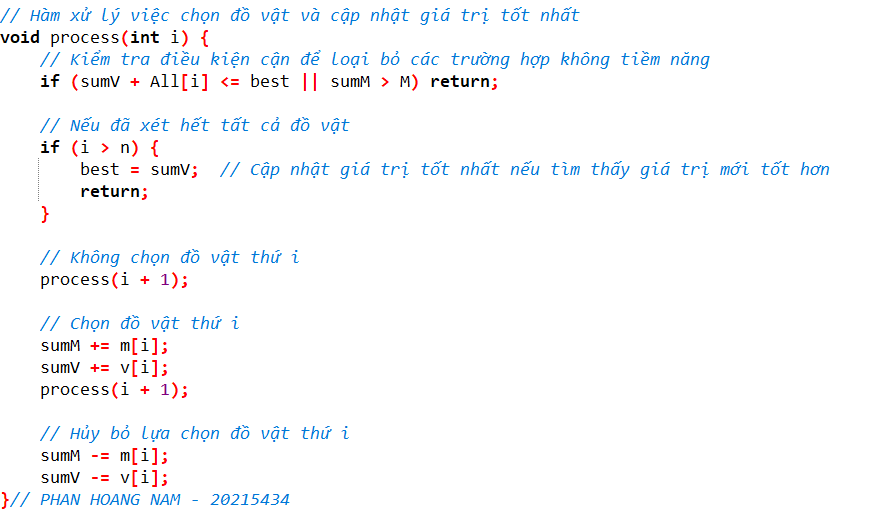
return 0;

}

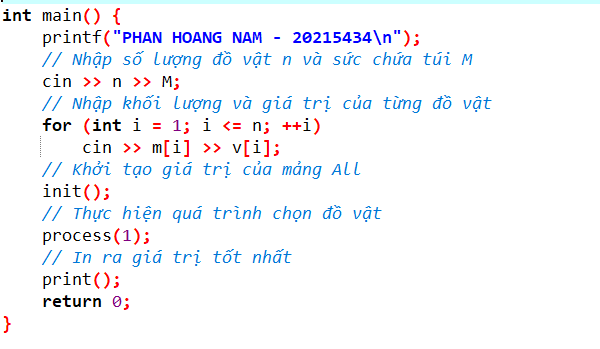
## **Bài 5.2:** Tìm và sửa các lỗi cú pháp Bài toán cái túi: Cho một cái túi có sức chứa M và n đồ vật. Đồ vật thứ i có khối lượng m i và giá trị v i . Cần chọn ra một số đồ vật để bỏ vào túi sao cho tổng khối lượng không quá M và tổng giá trị là lớn nhất có thể. Đoạn code sau đây giải bài toán cái túi bằng phương pháp duyệt nhánh cận. Hãy tìm và sửa các lỗi cú pháp



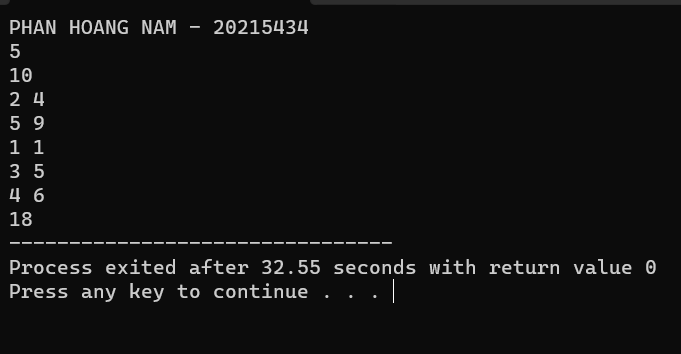
Hình 5. Code bài 5.2 ảnh 1



Hình 6. Code bài 5.2 ảnh 2



Hình 7. Code bài 5.2 ảnh 3



Hình 8. Output bài 5.2

// PHAN HOANG NAM - 20215434

/\*Bài tập 2: Tìm và sửa các lỗi cú pháp

Bài toán cái túi: Cho một cái túi có sức chứa M và n đồ vật. Đồ vật thứ i có khối lượng mi và giá trị vi.

Cần chọn ra một số đồ vật để bỏ vào túi sao cho tổng khối lượng không quá M và tổng giá trị là lớn nhất có thể.

Đoạn code sau đây giải bài toán cái túi bằng phương pháp duyệt nhánh cận. Hãy tìm và sửa các lỗi cú pháp

\*/

#include <iostream>

using namespace std;

int n, M, m[100], v[100];

int x[100], best, sumV, sumM, All[100];

// Hàm khởi tạo giá trị cho mảng All

void init() {

for (int i = n; i >= 1; --i) {

All[i] = All[i + 1] + v[i];

}

}

// Hàm in ra giá trị tốt nhất

void print() {

cout << best;

}// PHAN HOANG NAM - 20215434

// Hàm xử lý việc chọn đồ vật và cập nhật giá trị tốt nhất

void process(int i) {

// Kiểm tra điều kiện cận để loại bỏ các trường hợp không tiềm năng

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

// Nếu đã xét hết tất cả đồ vật

if (i > n) {

best = sumV; // Cập nhật giá trị tốt nhất nếu tìm thấy giá trị mới tốt hơn

return;

}

// Không chọn đồ vật thứ i

process(i + 1);

// Chọn đồ vật thứ i

sumM += m[i];

sumV += v[i];

process(i + 1);

// Hủy bỏ lựa chọn đồ vật thứ i

sumM -= m[i];

sumV -= v[i];

}// PHAN HOANG NAM - 20215434

int main() {

printf("PHAN HOANG NAM - 20215434\n");

// Nhập số lượng đồ vật n và sức chứa túi M

cin >> n >> M;

// Nhập khối lượng và giá trị của từng đồ vật

for (int i = 1; i <= n; ++i)

cin >> m[i] >> v[i];

// Khởi tạo giá trị của mảng All

init();

// Thực hiện quá trình chọn đồ vật

process(1);

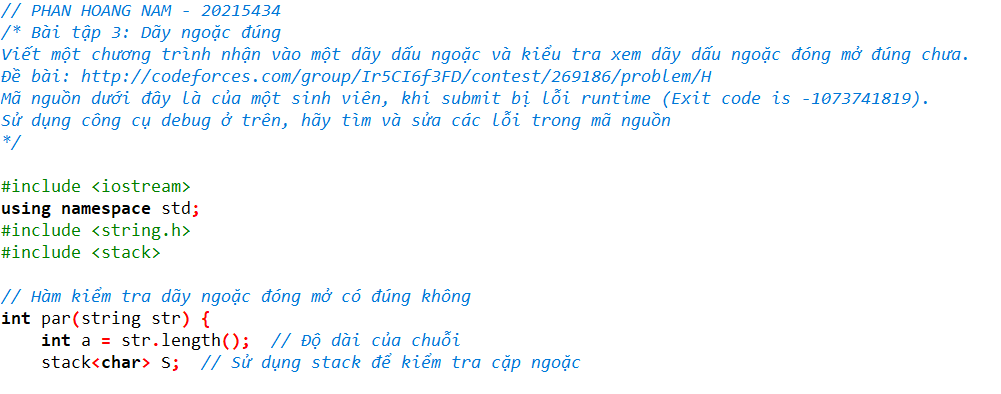
// In ra giá trị tốt nhất

print();

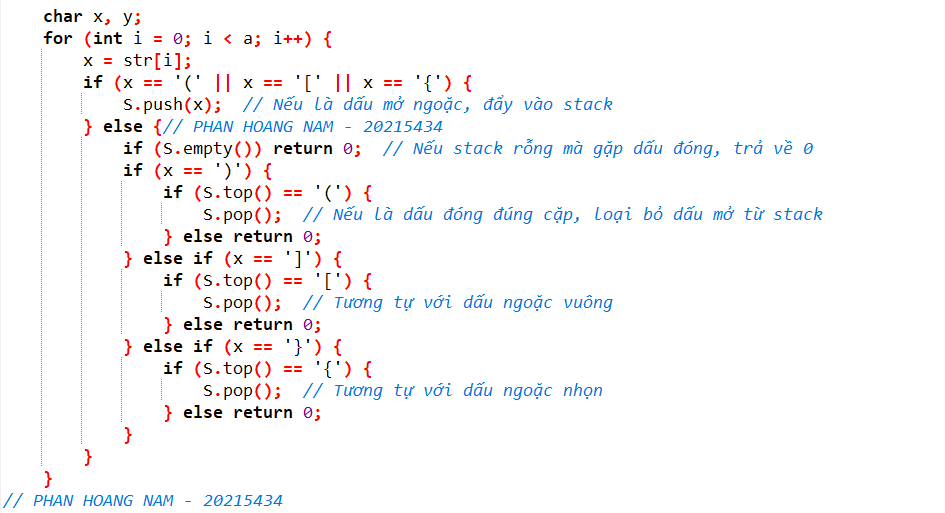
return 0;

}

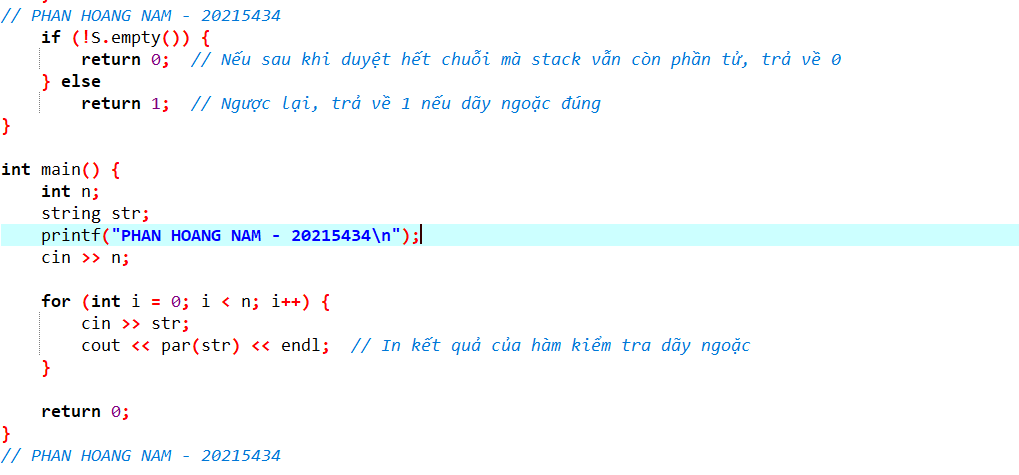
## **Bài 5.3:** Dãy ngoặc đúng Viết một chương trình nhận vào một dãy dấu ngoặc và kiểu tra xem dãy dấu ngoặc đóng mở đúng chưa. Đề bài: http://codeforces.com/group/Ir5CI6f3FD/contest/269186/problem/H Mã nguồn dưới đây là của một sinh viên, khi submit bị lỗi runtime (Exit code is -1073741819). Sử dụng công cụ debug ở trên, hãy tìm và sửa các lỗi trong mã nguồn



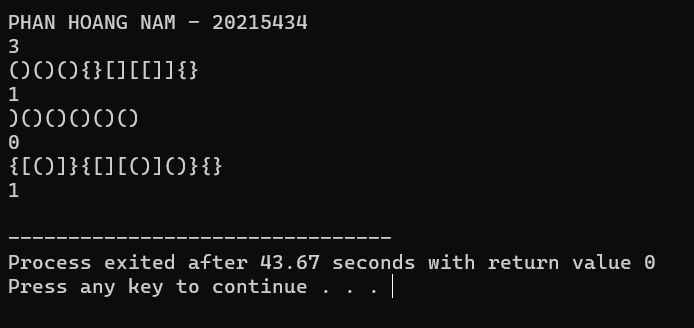
Hình 9. Code bài 5.3 ảnh 1



Hình 10. Code bài 5.3 ảnh 2



Hình 11. Code bài 5.3 ảnh 3



Hình 12. Output bài 5.3

// PHAN HOANG NAM - 20215434

/\* Bài tập 3: Dãy ngoặc đúng

Viết một chương trình nhận vào một dãy dấu ngoặc và kiểu tra xem dãy dấu ngoặc đóng mở đúng chưa.

Đề bài: http://codeforces.com/group/Ir5CI6f3FD/contest/269186/problem/H

Mã nguồn dưới đây là của một sinh viên, khi submit bị lỗi runtime (Exit code is -1073741819).

Sử dụng công cụ debug ở trên, hãy tìm và sửa các lỗi trong mã nguồn

\*/

#include <iostream>

using namespace std;

#include <string.h>

#include <stack>

// Hàm kiểm tra dãy ngoặc đóng mở có đúng không

int par(string str) {

int a = str.length(); // Độ dài của chuỗi

stack<char> S; // Sử dụng stack để kiểm tra cặp ngoặc

char x, y;

for (int i = 0; i < a; i++) {

x = str[i];

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

S.push(x); // Nếu là dấu mở ngoặc, đẩy vào stack

} else {// PHAN HOANG NAM - 20215434

if (S.empty()) return 0; // Nếu stack rỗng mà gặp dấu đóng, trả về 0

if (x == ')') {

if (S.top() == '(') {

S.pop(); // Nếu là dấu đóng đúng cặp, loại bỏ dấu mở từ stack

} else return 0;

} else if (x == ']') {

if (S.top() == '[') {

S.pop(); // Tương tự với dấu ngoặc vuông

} else return 0;

} else if (x == '}') {

if (S.top() == '{') {

S.pop(); // Tương tự với dấu ngoặc nhọn

} else return 0;

}

}

}

// PHAN HOANG NAM - 20215434

if (!S.empty()) {

return 0; // Nếu sau khi duyệt hết chuỗi mà stack vẫn còn phần tử, trả về 0

} else

return 1; // Ngược lại, trả về 1 nếu dãy ngoặc đúng

}

int main() {

int n;

string str;

printf("PHAN HOANG NAM - 20215434\n");

cin >> n;

for (int i = 0; i < n; i++) {

cin >> str;

cout << par(str) << endl; // In kết quả của hàm kiểm tra dãy ngoặc

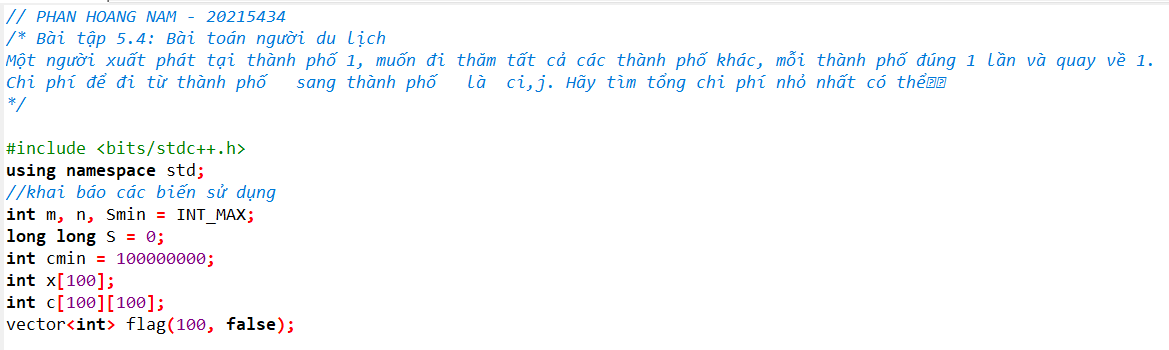
}

return 0;

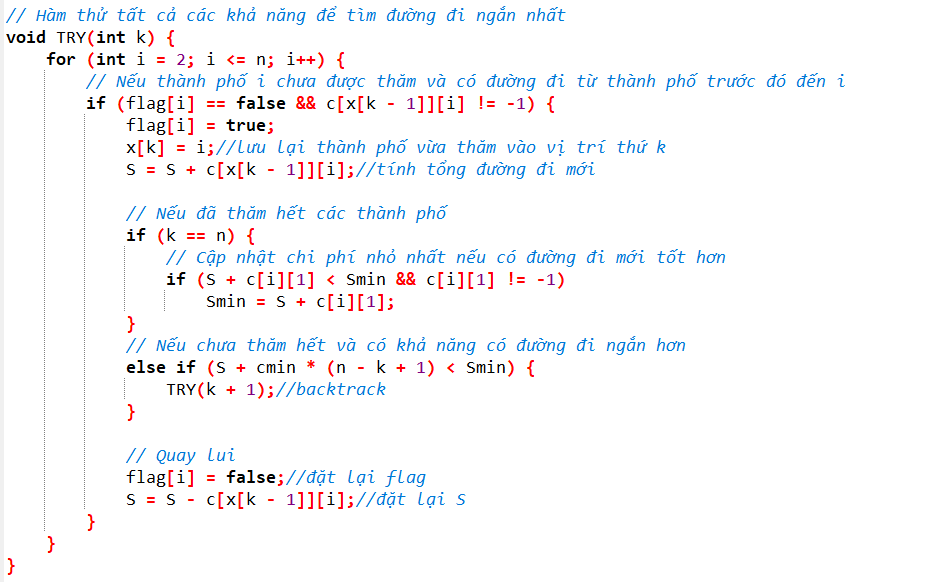
}

// PHAN HOANG NAM - 20215434

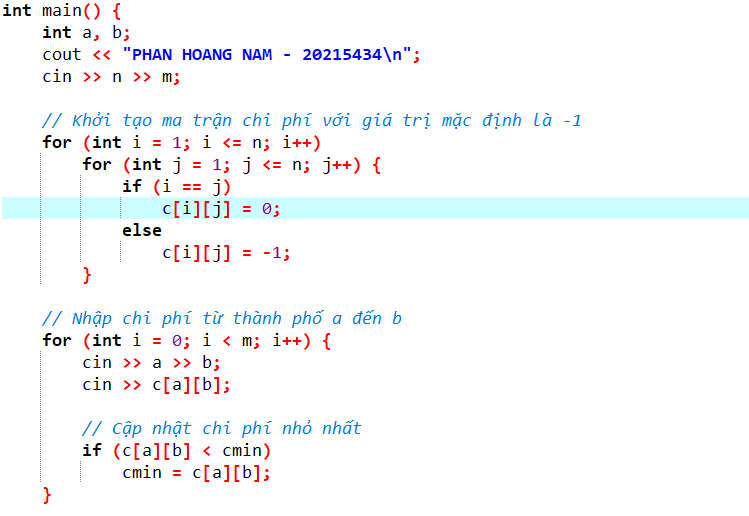
## **Bài 5.4:** Bài toán người du lịch Một người xuất phát tại thành phố 1, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng 1 lần và quay về 1. Chi phí để đi từ thành phố 𝑖 sang thành phố 𝑗 là ci,j. Hãy tìm tổng chi phí nhỏ nhất có thể



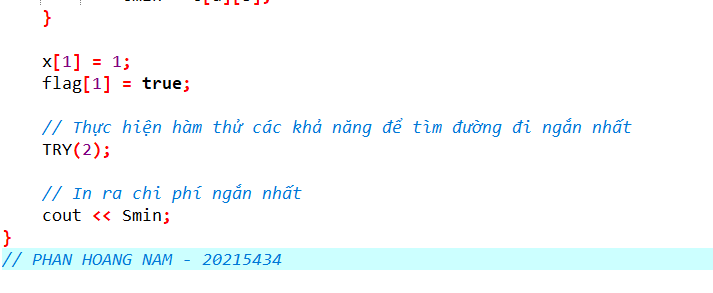
Hình 13. Code bài 5.4 ảnh 1



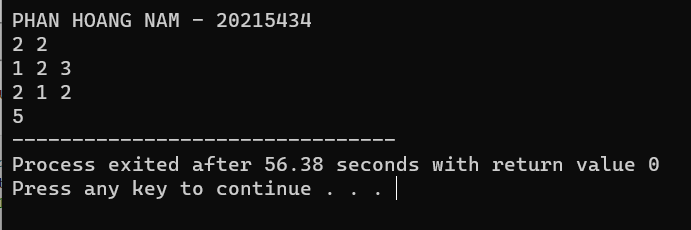
Hình 14. Code bài 5.4 ảnh 2



Hình 15. Code bài 5.4 ảnh 3



Hình 16. Code bài 5.4 ảnh 4



Hình 17. Output bài 5.4

// PHAN HOANG NAM - 20215434

/\* Bài tập 5.4: Bài toán người du lịch

Một người xuất phát tại thành phố 1, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng 1 lần và quay về 1.

Chi phí để đi từ thành phố 𝑖 sang thành phố 𝑗 là ci,j. Hãy tìm tổng chi phí nhỏ nhất có thể

\*/

#include <bits/stdc++.h>

using namespace std;

//khai báo các biến sử dụng

int m, n, Smin = INT\_MAX;

long long S = 0;

int cmin = 100000000;

int x[100];

int c[100][100];

vector<int> flag(100, false);

// Hàm thử tất cả các khả năng để tìm đường đi ngắn nhất

void TRY(int k) {

for (int i = 2; i <= n; i++) {

// Nếu thành phố i chưa được thăm và có đường đi từ thành phố trước đó đến i

if (flag[i] == false && c[x[k - 1]][i] != -1) {

flag[i] = true;

x[k] = i;//lưu lại thành phố vừa thăm vào vị trí thứ k

S = S + c[x[k - 1]][i];//tính tổng đường đi mới

// Nếu đã thăm hết các thành phố

if (k == n) {

// Cập nhật chi phí nhỏ nhất nếu có đường đi mới tốt hơn

if (S + c[i][1] < Smin && c[i][1] != -1)

Smin = S + c[i][1];

}

// Nếu chưa thăm hết và có khả năng có đường đi ngắn hơn

else if (S + cmin \* (n - k + 1) < Smin) {

TRY(k + 1);//backtrack

}

// Quay lui

flag[i] = false;//đặt lại flag

S = S - c[x[k - 1]][i];//đặt lại S

}

}

}

int main() {

int a, b;

cout << "PHAN HOANG NAM - 20215434\n";

cin >> n >> m;

// Khởi tạo ma trận chi phí với giá trị mặc định là -1

for (int i = 1; i <= n; i++)

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

if (i == j)

c[i][j] = 0;

else

c[i][j] = -1;

}

// Nhập chi phí từ thành phố a đến b

for (int i = 0; i < m; i++) {

cin >> a >> b;

cin >> c[a][b];

// Cập nhật chi phí nhỏ nhất

if (c[a][b] < cmin)

cmin = c[a][b];

}

x[1] = 1;

flag[1] = true;

// Thực hiện hàm thử các khả năng để tìm đường đi ngắn nhất

TRY(2);

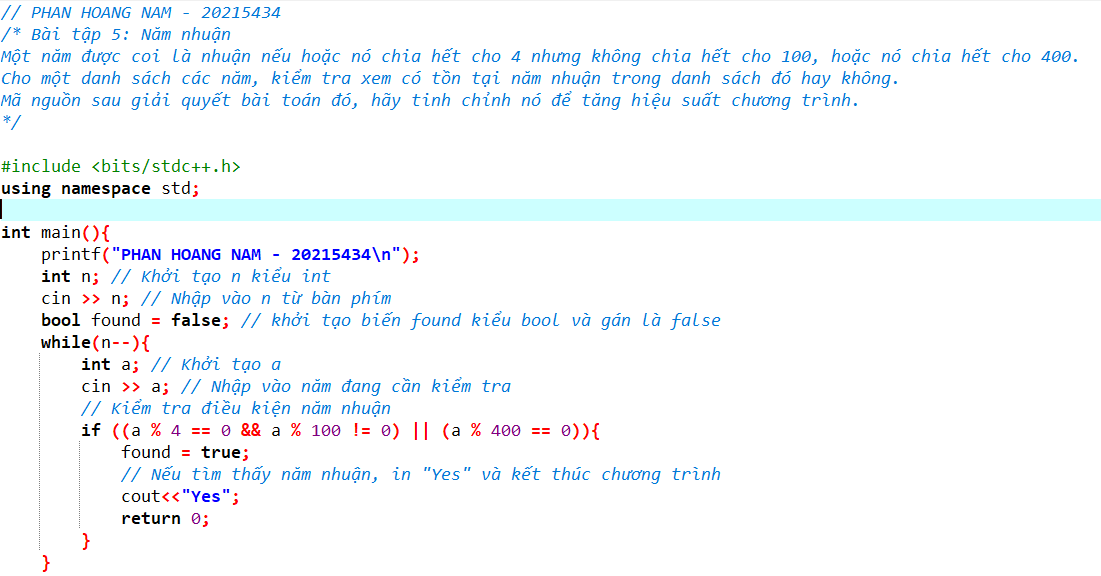
// In ra chi phí ngắn nhất

cout << Smin;

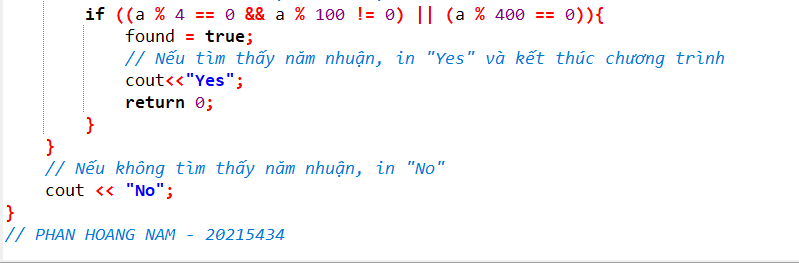
}

// PHAN HOANG NAM - 20215434

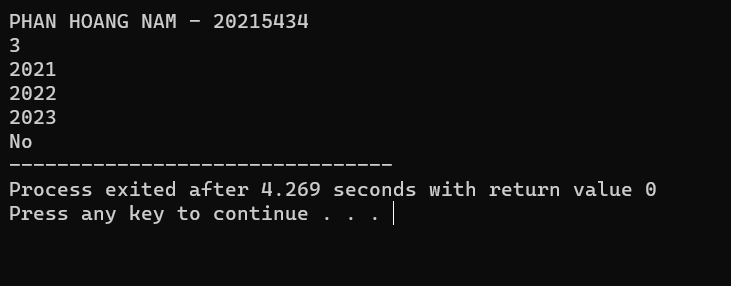
## **Bài 5.5:** Năm nhuận Một năm được coi là nhuận nếu hoặc nó chia hết cho 4 nhưng không chia hết cho 100, hoặc nó chia hết cho 400. Cho một danh sách các năm, kiểm tra xem có tồn tại năm nhuận trong danh sách đó hay không. Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.



Hình 18. Code bài 5.5 ảnh 1



Hình 19. Code bài 5.5 ảnh 2



Hình 20. Output bài 5.5

// PHAN HOANG NAM - 20215434

/\* Bài tập 5: Năm nhuận

Một năm được coi là nhuận nếu hoặc nó chia hết cho 4 nhưng không chia hết cho 100, hoặc nó chia hết cho 400.

Cho một danh sách các năm, kiểm tra xem có tồn tại năm nhuận trong danh sách đó hay không.

Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.

\*/

#include <bits/stdc++.h>

using namespace std;

int main(){

printf("PHAN HOANG NAM - 20215434\n");

int n; // Khởi tạo n kiểu int

cin >> n; // Nhập vào n từ bàn phím

bool found = false; // khởi tạo biến found kiểu bool và gán là false

while(n--){

int a; // Khởi tạo a

cin >> a; // Nhập vào năm đang cần kiểm tra

// Kiểm tra điều kiện năm nhuận

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

found = true;

// Nếu tìm thấy năm nhuận, in "Yes" và kết thúc chương trình

cout<<"Yes";

return 0;

}

}

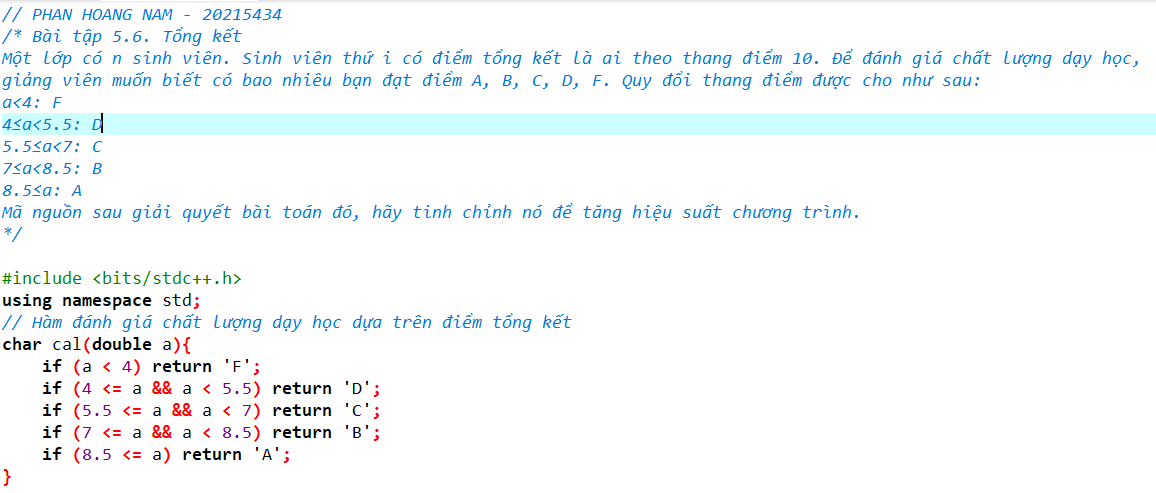
// Nếu không tìm thấy năm nhuận, in "No"

cout << "No";

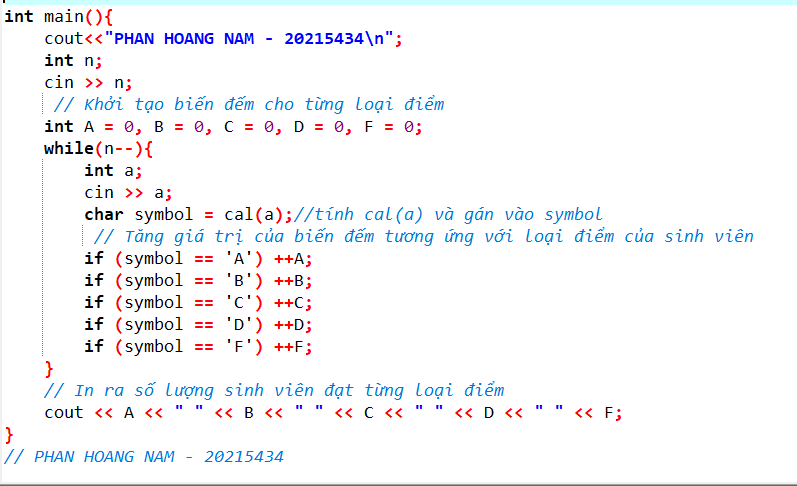
}

// PHAN HOANG NAM - 20215434

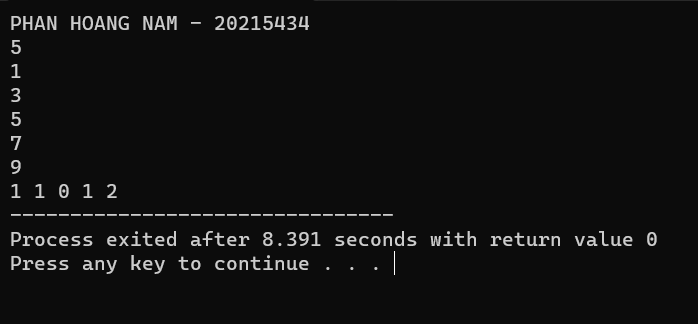
## **Bài 5.6:** Tổng kết Một lớp có n sinh viên. Sinh viên thứ i có điểm tổng kết là ai theo thang điểm 10. Để đánh giá chất lượng dạy học, giảng viên muốn biết có bao nhiêu bạn đạt điểm A, B, C, D, F. Quy đổi thang điểm được cho như sau: a<4: F 4≤a<5.5: D 5.5≤a<7: C 7≤a<8.5: B 8.5≤a: A Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.



Hình 22. Code bào 5.6 ảnh 1



Hình 23. Code bào 5.6 ảnh 2



Hình 24. Output bài 5.6

// PHAN HOANG NAM - 20215434

/\* Bài tập 5.6. Tổng kết

Một lớp có n sinh viên. Sinh viên thứ i có điểm tổng kết là ai theo thang điểm 10. Để đánh giá chất lượng dạy học,

giảng viên muốn biết có bao nhiêu bạn đạt điểm A, B, C, D, F. Quy đổi thang điểm được cho như sau:

a<4: F

4≤a<5.5: D

5.5≤a<7: C

7≤a<8.5: B

8.5≤a: A

Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.

\*/

#include <bits/stdc++.h>

using namespace std;

// Hàm đánh giá chất lượng dạy học dựa trên điểm tổng kết

char cal(double a){

if (a < 4) return 'F';

if (4 <= a && a < 5.5) return 'D';

if (5.5 <= a && a < 7) return 'C';

if (7 <= a && a < 8.5) return 'B';

if (8.5 <= a) return 'A';

}

int main(){

cout<<"PHAN HOANG NAM - 20215434\n";

int n;

cin >> n;

// Khởi tạo biến đếm cho từng loại điểm

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

while(n--){

int a;

cin >> a;

// Tăng giá trị của biến đếm tương ứng với loại điểm của sinh viên

if (cal(a) == 'A') ++A;

if (cal(a) == 'B') ++B;

if (cal(a) == 'C') ++C;

if (cal(a) == 'D') ++D;

if (cal(a) == 'F') ++F;

}

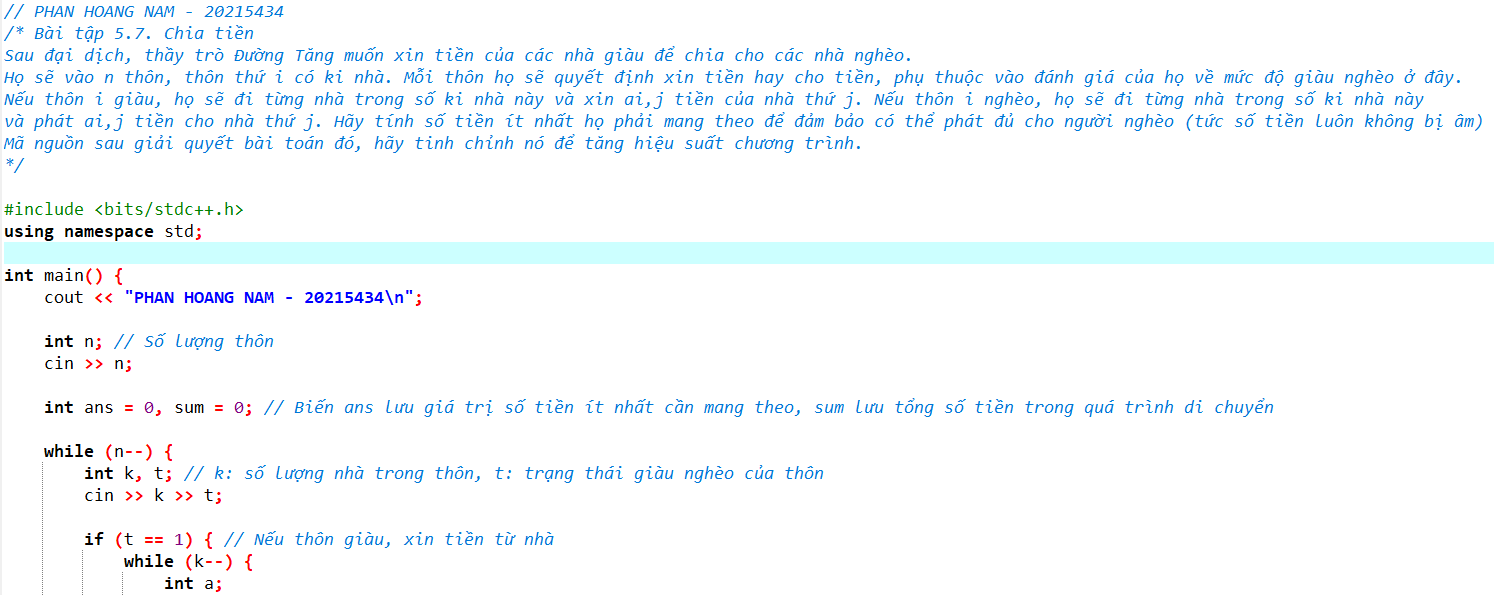
// In ra số lượng sinh viên đạt từng loại điểm

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

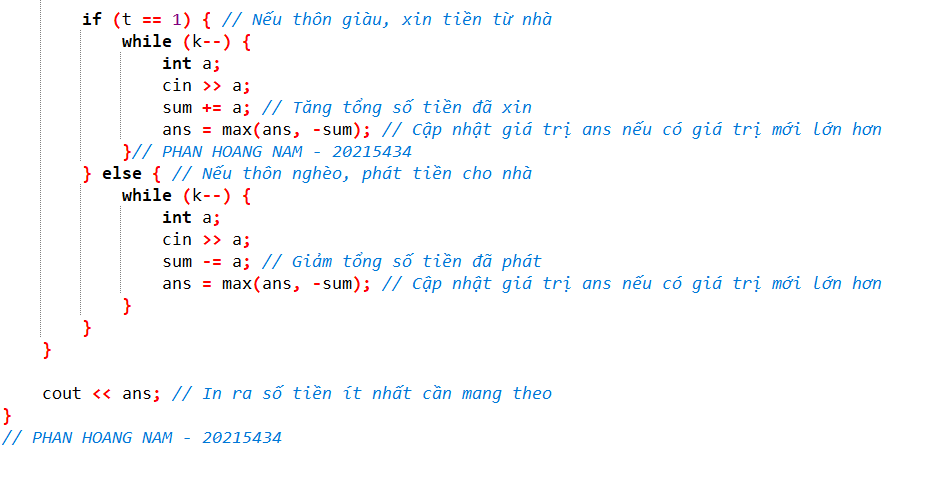
}

// PHAN HOANG NAM - 20215434

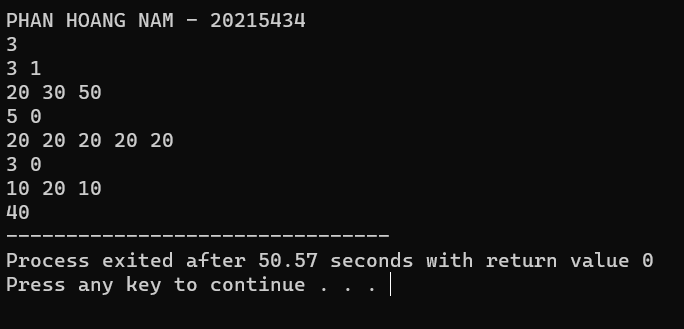
## **Bài 5.7**. Chia tiền Sau đại dịch, thầy trò Đường Tăng muốn xin tiền của các nhà giàu để chia cho các nhà nghèo. Họ sẽ vào n thôn, thôn thứ i có ki nhà. Mỗi thôn họ sẽ quyết định xin tiền hay cho tiền, phụ thuộc vào đánh giá của họ về mức độ giàu nghèo ở đây. Nếu thôn i giàu, họ sẽ đi từng nhà trong số ki nhà này và xin ai,j tiền của nhà thứ j. Nếu thôn i nghèo, họ sẽ đi từng nhà trong số ki nhà này và phát ai,j tiền cho nhà thứ j. Hãy tính số tiền ít nhất họ phải mang theo để đảm bảo có thể phát đủ cho người nghèo (tức số tiền luôn không bị âm) Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.



Hình 26. Code bài 5.7 ảnh 1



Hình 27. Code bài 5.7 ảnh 2



Hình 31. Output bài 5.7

// PHAN HOANG NAM - 20215434

/\* Bài tập 5.7. Chia tiền

Sau đại dịch, thầy trò Đường Tăng muốn xin tiền của các nhà giàu để chia cho các nhà nghèo.

Họ sẽ vào n thôn, thôn thứ i có ki nhà. Mỗi thôn họ sẽ quyết định xin tiền hay cho tiền, phụ thuộc vào đánh giá của họ về mức độ giàu nghèo ở đây.

Nếu thôn i giàu, họ sẽ đi từng nhà trong số ki nhà này và xin ai,j tiền của nhà thứ j. Nếu thôn i nghèo, họ sẽ đi từng nhà trong số ki nhà này

và phát ai,j tiền cho nhà thứ j. Hãy tính số tiền ít nhất họ phải mang theo để đảm bảo có thể phát đủ cho người nghèo (tức số tiền luôn không bị âm)

Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.

\*/

#include <bits/stdc++.h>

using namespace std;

int main() {

cout << "PHAN HOANG NAM - 20215434\n";

int n; // Số lượng thôn

cin >> n;

int ans = 0, sum = 0; // Biến ans lưu giá trị số tiền ít nhất cần mang theo, sum lưu tổng số tiền trong quá trình di chuyển

while (n--) {

int k, t; // k: số lượng nhà trong thôn, t: trạng thái giàu nghèo của thôn

cin >> k >> t;

if (t == 1) { // Nếu thôn giàu, xin tiền từ nhà

while (k--) {

int a;

cin >> a;

sum += a; // Tăng tổng số tiền đã xin

ans = max(ans, -sum); // Cập nhật giá trị ans nếu có giá trị mới lớn hơn

}// PHAN HOANG NAM - 20215434

} else { // Nếu thôn nghèo, phát tiền cho nhà

while (k--) {

int a;

cin >> a;

sum -= a; // Giảm tổng số tiền đã phát

ans = max(ans, -sum); // Cập nhật giá trị ans nếu có giá trị mới lớn hơn

}

}

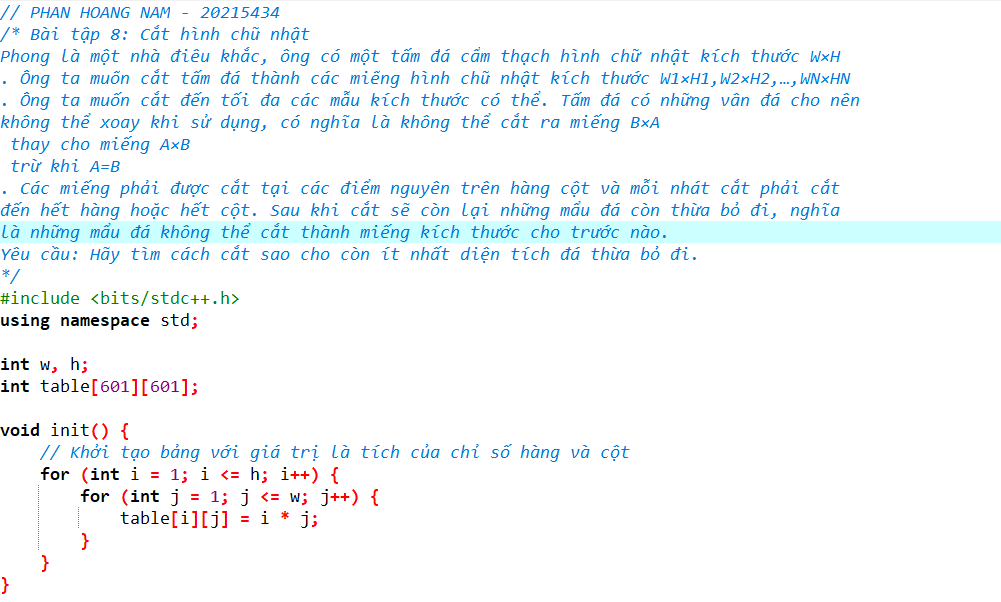
}

cout << ans; // In ra số tiền ít nhất cần mang theo

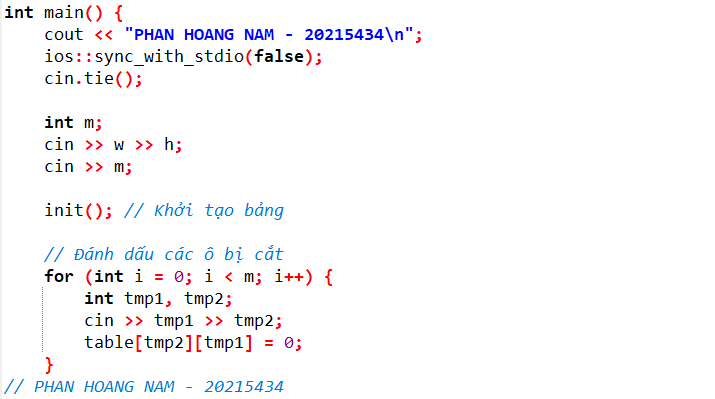
}

// PHAN HOANG NAM - 20215434

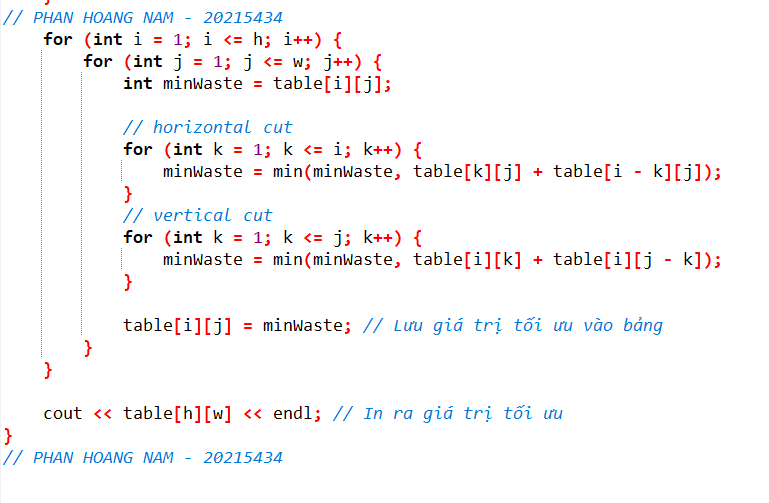
## **Bài 5.8:** Cắt hình chữ nhật Phong là một nhà điêu khắc, ông có một tấm đá cẩm thạch hình chữ nhật kích thước W×H . Ông ta muốn cắt tấm đá thành các miếng hình chữ nhật kích thước W1×H1,W2×H2,…,WN×HN . Ông ta muốn cắt đến tối đa các mẫu kích thước có thể. Tấm đá có những vân đá cho nên không thể xoay khi sử dụng, có nghĩa là không thể cắt ra miếng B×A thay cho miếng A×B trừ khi A=B . Các miếng phải được cắt tại các điểm nguyên trên hàng cột và mỗi nhát cắt phải cắt đến hết hàng hoặc hết cột. Sau khi cắt sẽ còn lại những mẩu đá còn thừa bỏ đi, nghĩa là những mẩu đá không thể cắt thành miếng kích thước cho trước nào. Yêu cầu: Hãy tìm cách cắt sao cho còn ít nhất diện tích đá thừa bỏ đi. Sử dụng công cụ debug ở trên, hãy tìm và sửa các lỗi trong mã nguồn dưới đây



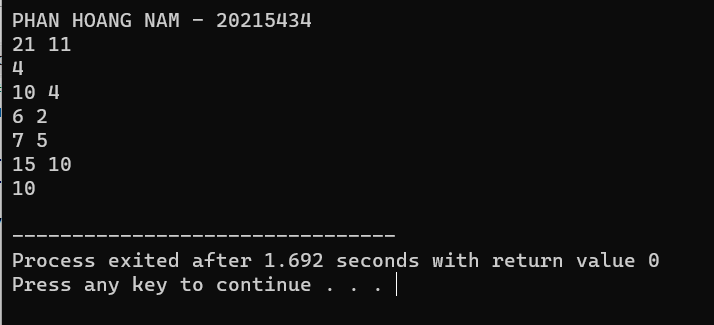
Hình 32. Code bài 5.8 ảnh 1



Hình 33. Code bài 5.8 ảnh 2



Hình 34. Code bài 5.8 ảnh 3



Hình 37. Output bài 5,8

// PHAN HOANG NAM - 20215434

/\* Bài tập 8: Cắt hình chữ nhật

Phong là một nhà điêu khắc, ông có một tấm đá cẩm thạch hình chữ nhật kích thước W×H

. Ông ta muốn cắt tấm đá thành các miếng hình chữ nhật kích thước W1×H1,W2×H2,…,WN×HN

. Ông ta muốn cắt đến tối đa các mẫu kích thước có thể. Tấm đá có những vân đá cho nên

không thể xoay khi sử dụng, có nghĩa là không thể cắt ra miếng B×A

thay cho miếng A×B

trừ khi A=B

. Các miếng phải được cắt tại các điểm nguyên trên hàng cột và mỗi nhát cắt phải cắt

đến hết hàng hoặc hết cột. Sau khi cắt sẽ còn lại những mẩu đá còn thừa bỏ đi, nghĩa

là những mẩu đá không thể cắt thành miếng kích thước cho trước nào.

Yêu cầu: Hãy tìm cách cắt sao cho còn ít nhất diện tích đá thừa bỏ đi.

\*/

#include <bits/stdc++.h>

using namespace std;

int w, h;

int table[601][601];

void init() {

// Khởi tạo bảng với giá trị là tích của chỉ số hàng và cột

for (int i = 1; i <= h; i++) {

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

table[i][j] = i \* j;

}

}

}

int main() {

cout << "PHAN HOANG NAM - 20215434\n";

ios::sync\_with\_stdio(false);

cin.tie();

int m;

cin >> w >> h;

cin >> m;

init(); // Khởi tạo bảng

// Đánh dấu các ô bị cắt

for (int i = 0; i < m; i++) {

int tmp1, tmp2;

cin >> tmp1 >> tmp2;

table[tmp2][tmp1] = 0;

}

// PHAN HOANG NAM - 20215434

for (int i = 1; i <= h; i++) {

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

int minWaste = table[i][j];

// horizontal 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; // Lưu giá trị tối ưu vào bảng

}

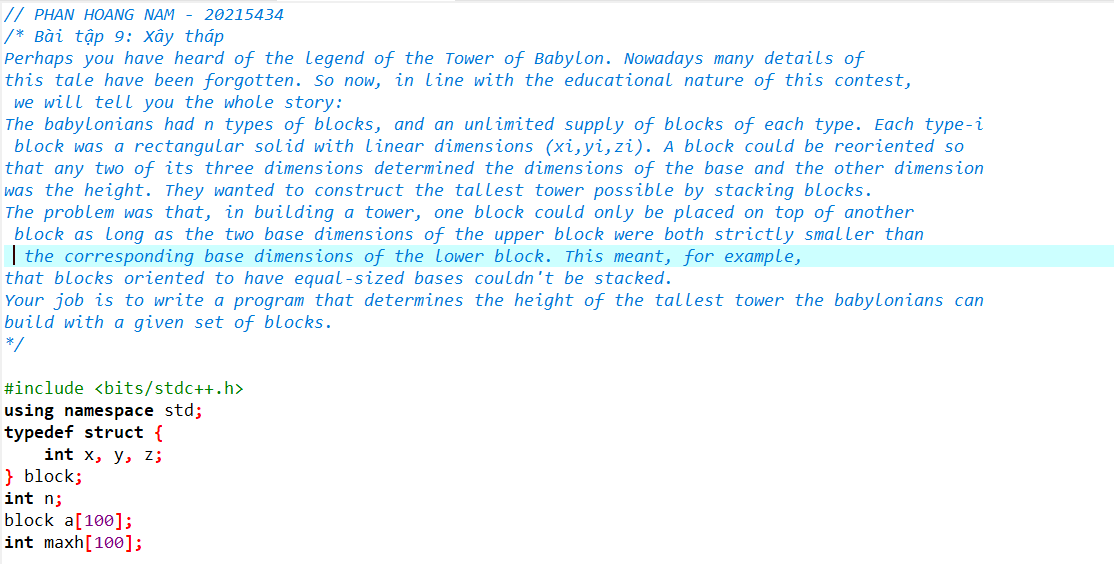
}

cout << table[h][w] << endl; // In ra giá trị tối ưu

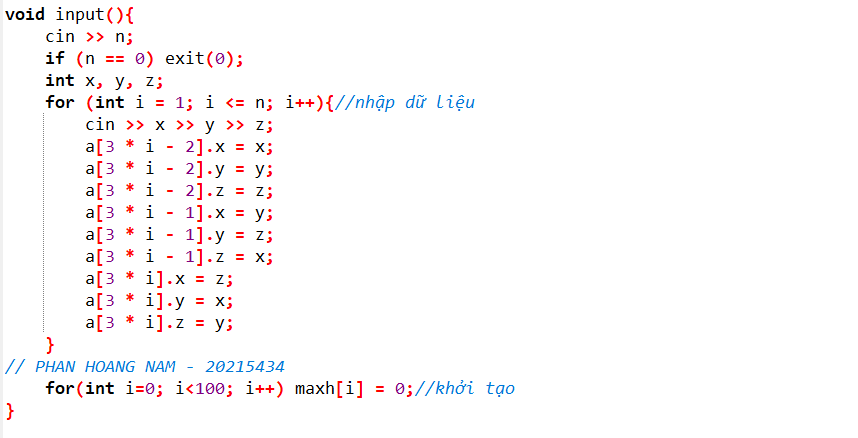
}

// PHAN HOANG NAM - 20215434

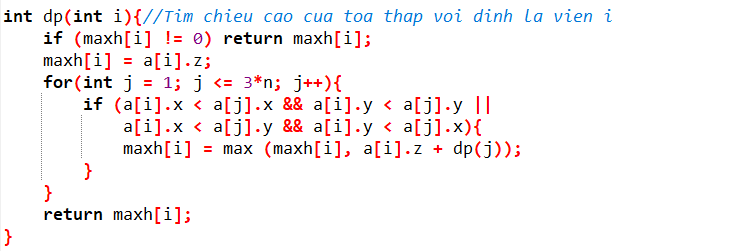
## **Bài 5.9:** Xây tháp Perhaps you have heard of the legend of the Tower of Babylon. Nowadays many details of this tale have been forgotten. So now, in line with the educational nature of this contest, we will tell you the whole story: The babylonians had n types of blocks, and an unlimited supply of blocks of each type. Each type-i block was a rectangular solid with linear dimensions (xi,yi,zi). A block could be reoriented so that any two of its three dimensions determined the dimensions of the base and the other dimension was the height. They wanted to construct the tallest tower possible by stacking blocks. The problem was that, in building a tower, one block could only be placed on top of another block as long as the two base dimensions of the upper block were both strictly smaller than the corresponding base dimensions of the lower block. This meant, for example, that blocks oriented to have equal-sized bases couldn't be stacked. Your job is to write a program that determines the height of the tallest tower the babylonians can build with a given set of blocks.



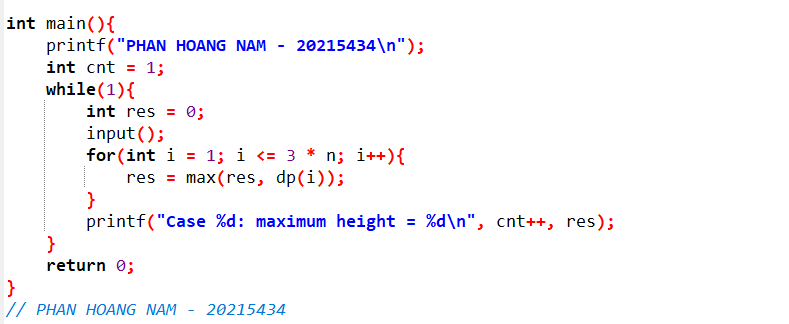
Hình 39. Code bài 5.9 ảnh 1



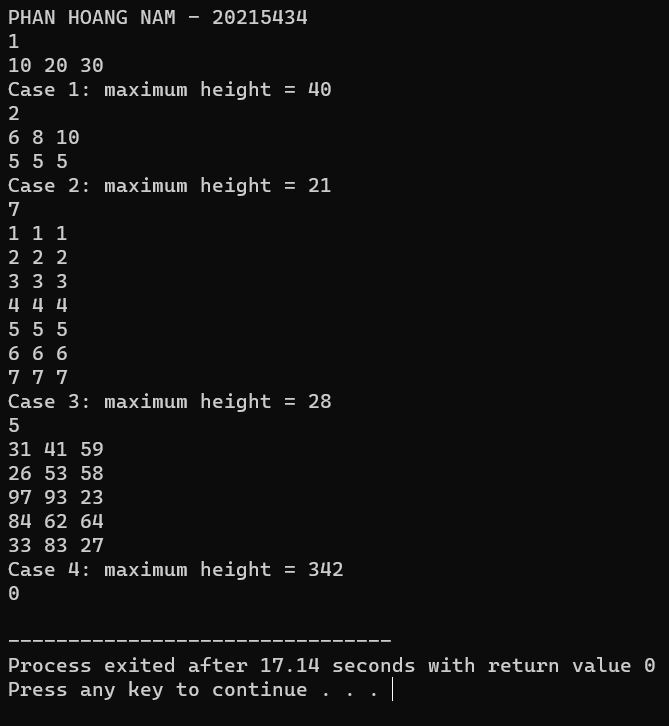
Hình 40. Code bài 5.9 ảnh 2



Hình 41. Code bài 5.9 ảnh 3



Hình 42. Code bài 5.9 ảnh 4



Hình 43. Output bài 5.9

// PHAN HOANG NAM - 20215434

/\* Bài tập 9: Xây tháp

Perhaps you have heard of the legend of the Tower of Babylon. Nowadays many details of

this tale have been forgotten. So now, in line with the educational nature of this contest,

we will tell you the whole story:

The babylonians had n types of blocks, and an unlimited supply of blocks of each type. Each type-i

block was a rectangular solid with linear dimensions (xi,yi,zi). A block could be reoriented so

that any two of its three dimensions determined the dimensions of the base and the other dimension

was the height. They wanted to construct the tallest tower possible by stacking blocks.

The problem was that, in building a tower, one block could only be placed on top of another

block as long as the two base dimensions of the upper block were both strictly smaller than

the corresponding base dimensions of the lower block. This meant, for example,

that blocks oriented to have equal-sized bases couldn't be stacked.

Your job is to write a program that determines the height of the tallest tower the babylonians can

build with a given set of blocks.

\*/

#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++){//nhập dữ liệu

cin >> x >> y >> z;

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;

}

// PHAN HOANG NAM - 20215434

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

}

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));

}

}

return maxh[i];

}

int main(){

printf("PHAN HOANG NAM - 20215434\n");

int cnt = 1;

while(1){

int res = 0;

input();

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

res = max(res, dp(i));

}

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

}

return 0;

}

// PHAN HOANG NAM - 20215434