MỤC LỤC

Bài thực hành số 5 – Tuần 16………………………………………………………………..02

Bài 5.1: Đoạn code sau liệt kê tất cả các hoán vị n số. Hãy tìm và sửa các lỗi cú pháp như hướng dẫn……...…….…………...……………………………………….………………….02

Bài 5.2: 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.……………………...……..03

Bài 5.3: Viết một chương trình nhận vào một dãy dấu ngoặc và kiểu tra xem dãy dấu ngoặc đóng mở đúng chưa...……….…………………………………………………...…..……….05

Bài 5.4: Bài toán người du lịch…………………………………….………………………...07

Bài 5.5: 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..………………………….……………………………11

Bài 5.6: 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.……………………………………………………………………...…...………………….12

Bài 5.7: 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)…...……...……………………………...…………………….14

Bài 5.8: Cắt hình chữ nhật………………………….……………………………..………….15

Bài 5.9: Xây tháp.……………………………………………………………..……………...18

BÀI THỰC HÀNH SỐ 5 – TUẦN 16

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

*#include <stdio.h>*

int x[100], mark[100], n;

void print(){

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

print("\n");

}

void process(int i) {

**if** (i **>** n){

printf();

**return**;

}

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

**if** (**!**mark**[**j**]){**

mark[j] **=** 1;

x[i] **=** j;

process(i**+**1);

mark[j] **=** 0;

}

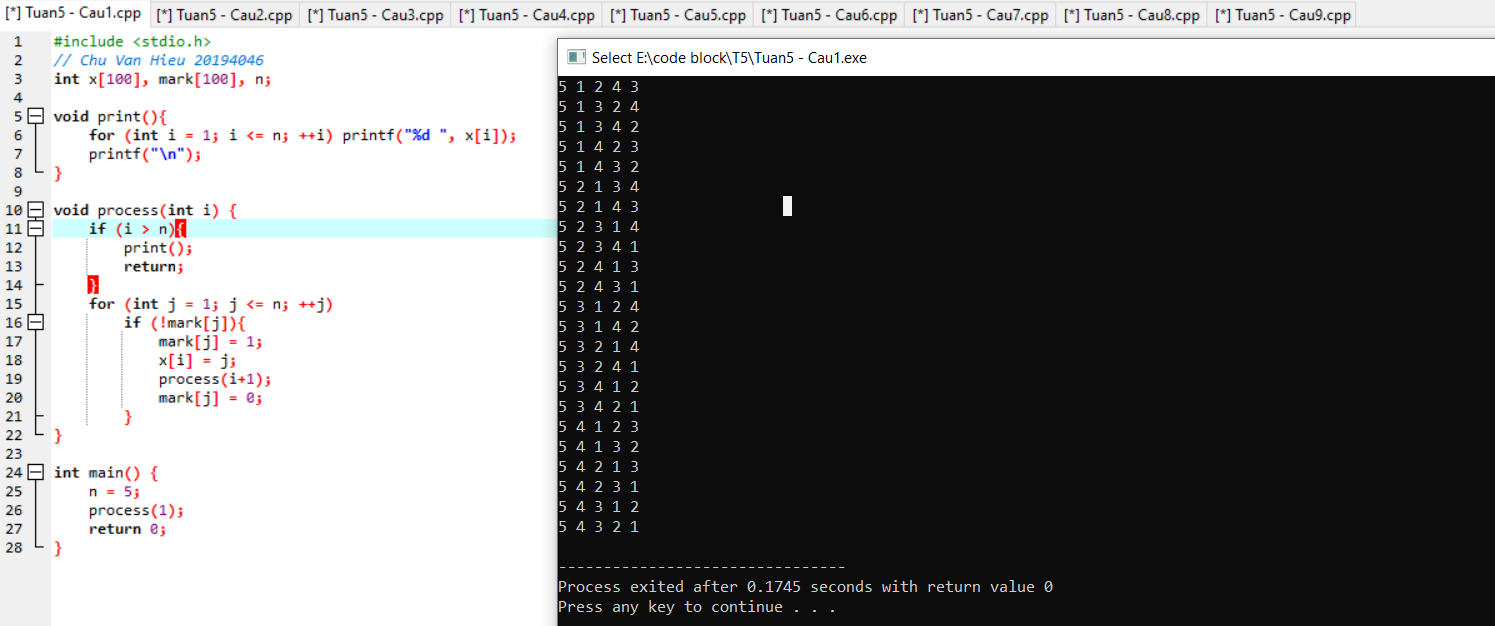
}

int main() {

n **=** 5;

process(1);

**return** 0;

}

Bài 5.2: 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];

void init(){

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

all[i] **=** all[i**+**1] **+** v[i];

}

}

void print() {

cout **<<** best;

}

void process(int i){

**if** (sumV **+** all[i] **<=** best **||** sumM **>** M) **return** 0;

**if** (i **>** n){

best **=** sumV

**return** 0;

}

process(i**+**1);

sumM **+=** m[i];

sumV **+=** v[i];

process(i**+**1);

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;

}

Ảnh có chứa văn bản

Mô tả được tạo tự động

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.

*#include <iostream>*

using namespace std;

*#include <string.h>*

*#include <stack>*

int par(string str){

int a **=** str**.**length();

stack**<**char**>** S;

char x, y;

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

x **=** str[i];

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

S**.**push(x);

}

**else** {

**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){

**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;

}

Ảnh có chứa văn bản

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự động

Bài 5.4: Bài toán người du lịch

Ảnh có chứa văn bản

Mô tả được tạo tự động

Dưới đây là solution của một bạn sinh viên, khi submit bị sai kết quả. Hãy sử dụng hướng dẫn phía trên và thuật toán trực tiếp (được cho phía dưới) để tìm ra một test sai.

*#include <bits/stdc++.h>*

using namespace std;

int m, n, Smin **=** 100000;

long long S **=** 0;

int cmin **=** 100000000;

int x[100];

int c[100][100];

vector**<**int**>** flag(100, false);

void TRY(int k)

{

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

{

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

{

flag[i] **=** true;

x[k] **=** i;

S **=** S **+** c[x[k **-** 1]][i];

**if** (k **==** n)

{

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

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

}

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

{

TRY(k **+** 1);

}

flag[i] **=** false;

S **=** S **-** c[x[k **-** 1]][i];

}

}

}

main()

{

int a, b;

cin **>>** n **>>** m;

**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;

}

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

{

cin **>>** a **>>** b;

cin **>>** c[a][b];

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

cmin **=** c[a][b];

}

x[1] **=** 1;

flag[1] **=** true;

TRY(2);

cout **<<** Smin;

}

/\*

*#include <bits/stdc++.h>*

using namespace std;

int main(){**//**code trau

int n, m;

cin **>>** n **>>** m;

vector**<**vector**<**int**>>** c(n, vector**<**int**>**(n, 1e8));

**while** (m**--**){

int x, y, w;

cin **>>** x **>>** y **>>** w;

c[x**-**1][y**-**1] **=** min(c[x**-**1][y**-**1], w);

}

int ans **=** 1e9;

vector**<**int**>** p(n);

iota(p**.**begin(), p**.**end(), 0);

do{

int sum **=** 0;

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

sum **+=** c[p[i]][p[(i**+**1)**%n**]];

ans **=** min(ans, sum);

} **while**(next\_permutation(p**.**begin()**+**1, p**.**end()));

cout **<<** ans **<<** endl;

}

\*/

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự động

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.

*#include <bits/stdc++.h>*

using namespace std;

int main(){

int n;

cin **>>** n;

bool found **=** false;

**while**(n**--**){

int a;

cin **>>** a;

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

found **=** true;

}

**if** (found) cout **<<** "Yes";

**else** cout **<<** "No";

}

Ảnh có chứa văn bản

Mô tả được tạo tự động

Bài 5.6: 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;

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(){

int n;

cin **>>** n;

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

**while**(n**--**){

int a;

cin **>>** a;

**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;

}

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

}

*#include <bits/stdc++.h>*

using namespace std;

int main(){

int n;

cin **>>** n;

int ans **=** 0, sum **=** 0;

**while**(n**--**){

int k, t;

cin **>>** k **>>** t;

**while**(k**--**){

int a;

cin **>>** a;

**if** (t **==** 1) sum **+=** a;

**if** (t **==** **-**1) sum **-=** a;

ans **=** max(ans, **-**sum);

}

}

cout **<<** ans;

}

Ảnh có chứa văn bản

Mô tả được tạo tự động

Bài 5.7: 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(){

int n;

cin **>>** n;

int ans **=** 0, sum **=** 0;

**while**(n**--**){

int k, t;

cin **>>** k **>>** t;

**while**(k**--**){

int a;

cin **>>** a;

**if** (t **==** 1) sum **+=** a;

**if** (t **==** **-**1) sum **-=** a;

ans **=** max(ans, **-**sum);

}

}

cout **<<** ans;

}

Ảnh có chứa văn bản

Mô tả được tạo tự động

Bài 5.8: Cắt hình chữ nhật

Đề bài: <http://codeforces.com/group/Ir5CI6f3FD/contest/276073/problem/G>

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.

**Input**

Dòng đầu tiên chứa hai số nguyên: W và H.

Dòng thứ hai chứa một số nguyên N. N dòng tiếp theo mỗi dòng chứa hai số nguyên Wi và Hi.

**Output**

Kết quả ghi ra duy nhất một số nguyên là tổng diện tích nhỏ nhất các miếng thừa bỏ đi.

**Scoring**

* 1 ≤ W ≤ 600, 1 ≤ H ≤ 600, 0 < N ≤ 200, 1 ≤ Wi ≤ W and 1 ≤ Hi ≤ H.
* Có 50% số test ứng với W ≤ 20, H ≤ 20 và N ≤ 5.

*#include <bits/stdc++.h>*

using namespace std;

int w, h;

int table[601][601];

void init() {

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

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

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

}

}

}

int main()

{

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

cin**.**tie();

int w, h, m;

cin **>>** w **>>** h;

cin **>>** m;

init();

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

int tmp1, tmp2;

cin **>>** tmp1 **>>** tmp2;

table[tmp2][tmp1] **=** 0;

}

**//**dp

**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**<=**i; k**++**) {

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

}

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

}

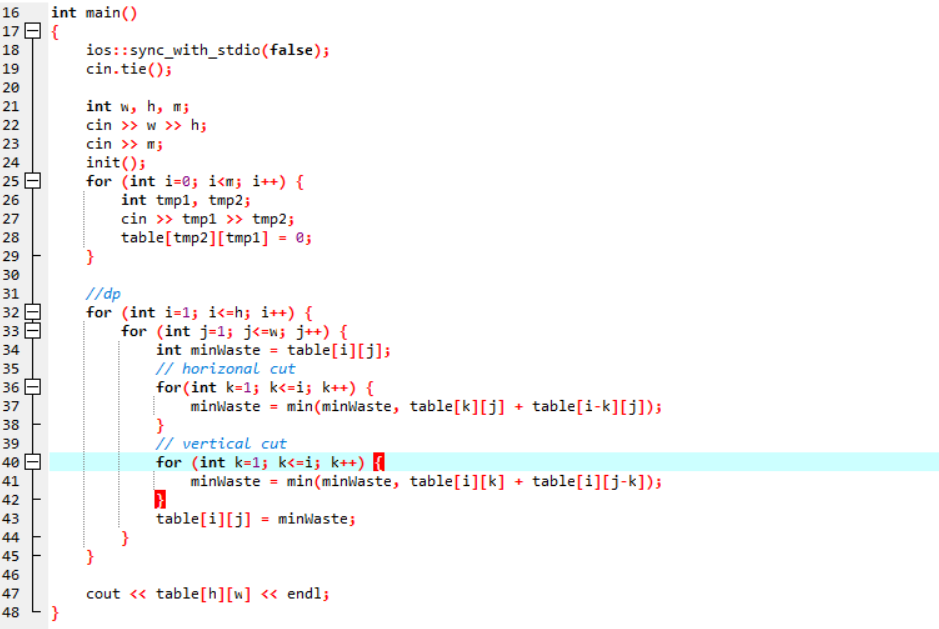
}

cout **<<** table[h][w] **<<** endl;

}

Ảnh có chứa văn bản

Mô tả được tạo tự động



Bài 5.9: Xây tháp

Đề bài: <http://codeforces.com/group/Ir5CI6f3FD/contest/276073/problem/I>

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.

**Input**

The input file will contain one or more test cases. The first line of each test case contains an integer n, representing the number of different blocks in the following data set. The maximum value for n is 30. Each of the next n lines contains three integers representing the values xi, yi and zi.

Input is terminated by a value of zero (0) for n.

**Output**

For each test case, print one line containing the case number (they are numbered sequentially starting from 1) and the height of the tallest possible tower in the format "Case *case*: maximum height = height"

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động