**Bài thực hành 5: Thực hành gỡ rối, kiểm thử và tinh chỉnh mã nguồn**

**Phần 1. Thực hành gỡ rối**

**1.1 Hướng dẫn tìm một số lỗi cú pháp (Syntax Error) thường gặp**

Lỗi cú pháp còn gọi là lỗi biên dịch (Compiler error). Đây là loại lỗi sơ đẳng nhất trong lập trình. Lỗi thường xảy ra do lập trình viên gõ sai cấu trúc của ngôn ngữ (ví dụ như thiếu dấu kết thúc một câu lệnh, một số ngôn ngữ phân biệt từ khoá với chữ hoa, chữ thường thì lại gõ chữ hoa...). Chương trình sẽ không thể biên địch được khi gặp lỗi này và thông báo lỗi. Các trình soạn thảo (Editor) hoặc các IDE tiên tiến hiện nay như CodeBlocks, Visual Studio, Eclipse, Android Studio, NetBean... đều hỗ trợ cho lập trình viên một cách trực quan để xử lý lỗi này. Phần này sẽ hướng dẫn các bạn sử dụng trình báo lỗi của CodeBlocks để sửa các lỗi cú pháp.

**Lỗi 1: Thiếu dấu chấm phẩy**

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Lỗi 2: Gõ sai tên biến/hàm**

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Lỗi 3: Thiếu dấu ngoặc**

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Lỗi 4: Đặt tên trùng với từ khóa**

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Lỗi 5: Sử dụng sai các hàm có sẵn**

Ảnh có chứa văn bản

Mô tả được tạo tự động Ảnh có chứa văn bản

Mô tả được tạo tự động

**1.2 Thực hành sửa lỗi cú pháp**

**Bài tập 1: Tìm và sửa các lỗi cú pháp**

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

In [ ]:

*#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 tập 2: Tìm và sửa các lỗi cú pháp**

Bài toán cái túi: Cho một cái túi có sức chứa MM và nn đồ vật. Đồ vật thứ ii có khối lượng mimi và giá trị vivi. Cần chọn ra một số đồ vật để bỏ vào túi sao cho tổng khối lượng không quá MM 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

In [ ]:

*#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();

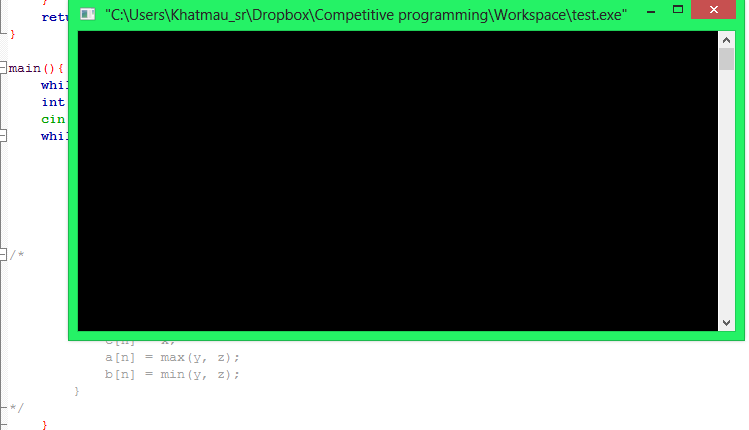
**return** 0;

}

**1.3 Hướng dẫn tìm một số lỗi thực thi (Runtime Error)**

Xảy ra bất ngờ khi chương trình đang chạy. Loại lỗi này thường xảy ra do người lập trình viết code không cẩn thận, không lường hết các trường hợp xảy ra, khiến chương trình đang chạy thì bị lỗi treo màn hình hoặc thoát hẳn ra khỏi chương trình... Lỗi này có thể phát hiện bằng cách Debug.

**Lỗi 1: Lặp vô hạn**

Lỗi này thường gặp khi dùng các lệnh lặp với số lần chưa biết trước, ví dụ lệnh while. Biểu hiện của lỗi này là chương trình sẽ không dừng (như hình sau) nhưng cũng không sinh lỗi. 

**Lỗi 2: Lỗi tràn bộ nhớ đệm (stack overflow)**

Lỗi này thường gặp khi gọi đệ quy quá sâu, thường do thiếu neo đệ quy dẫn đến độ sâu vô hạn. Biểu hiện của lỗi này là chương trình chạy sinh lỗi với mã lỗi âm (như hình sau). Ảnh có chứa văn bản

Mô tả được tạo tự động

**1.4 Hướng dẫn tìm một số lỗi logic**

Các loại lỗi tràn mảng và tràn số đều là undefined behavior, C++ sẽ không sinh lỗi khi chạy nhưng những lỗi này khiến kết quả chạy bị sai

**Lỗi 1: Lỗi tràn mảng**

Lỗi này gặp phải khi truy xuất ngoài phạm vi khai báo của mảng / vector. Ảnh có chứa văn bản

Mô tả được tạo tự động

**Lỗi 2: Lỗi tràn số**

Lỗi này gặp phải khi phạm vi tính toán hoặc lưu trữ nằm ngoài kiểu dữ liệu của biến. Ảnh có chứa văn bản

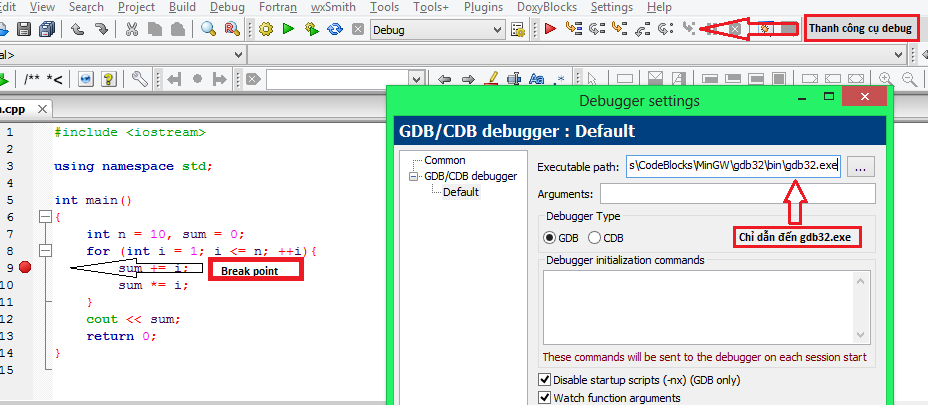
Mô tả được tạo tự động

**1.5 Tìm vị trí lỗi bằng trình debug**

CodeBlocks cung cấp công cụ debug bằng cách chạy từng dòng lệnh. Để sử dụng tính năng này, bạn cần tạo project thay vì tạo một file .cpp riêng lẻ

Tài liệu tham khảo: <http://wiki.codeblocks.org/index.php/Debugging_with_Code::Blocks>

**1: Giới thiệu**

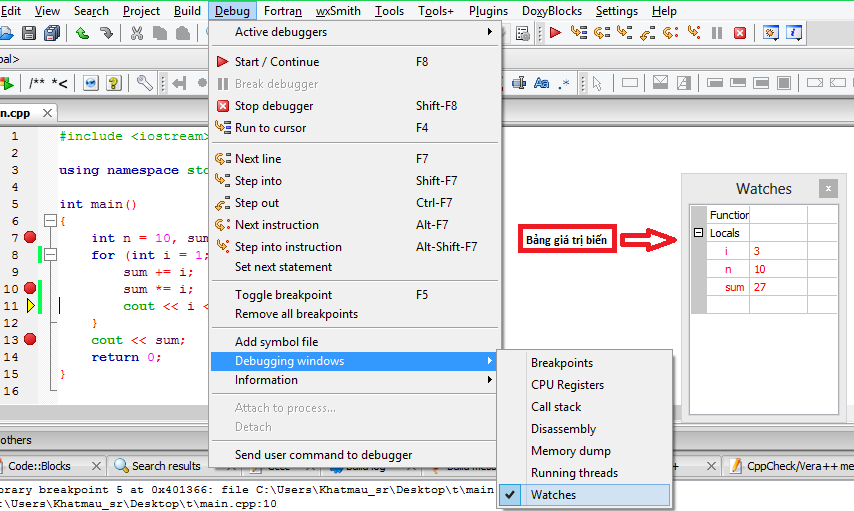
Để sử dụng trình debug của codeblocks, các bạn cần tạo project, cài đặt môi trường cho trình dịch debug như hình 

**2: Sử dụng chức năng nhảy lệnh**

Trình debug của codeblocks cung cấp công cụ cho phép đưa con trỏ lệnh đến một số vị trí định trước, bao gồm: Nhảy đến dòng lệnh tiếp theo, nhảy đến breakpoint tiếp theo, nhảy qua (hoặc ra khỏi) một khối lệnh/chương trình con, nhảy vào một chương trình con (thay vì xem nó như một dòng lệnh). Ảnh có chứa văn bản

Mô tả được tạo tự động

**3: Sử dụng chức năng hiển thị giá trị biến**

Đây là chức năng rất đắc lực, bạn có thể xem giá trị các biến khi chương trình dừng ở một dòng lệnh nhất định nào đó 

**4: Breakpoint nâng cao**

Có thể đặt điều kiện để một dòng trở thành breakpoint, khi điều kiện thỏa mãn thì dòng lệnh này sẽ trở thành breakpoint. Click chuột phải vào breakpoint để cài đặt Ảnh có chứa văn bản

Mô tả được tạo tự động

**1.6 Tìm vị trí lỗi bằng cách in ra màn hình / file log**

Không phụ thuộc vào IDE, không cần chạy từng lệnh, việc debug bằng cách thêm các lệnh in ra màn hình / file log được dùng khá phổ biến, đặc biệt với các project lớn và phức tạp. Cách làm rất đơn giản: Thêm các lệnh in giá trị các biến mà chúng ta muốn biết, sau đó chạy chương trình và tìm lỗi dựa vào những thông tin được in ra (hình minh họa).

In [ ]:

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

using namespace std;

int main(int argc, char**\*\*** argv){

int n;

cin **>>** n;

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

**for** (int **&**x : a) cin **>>** x;

printf("Done input\n");

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

**for** (int i **=** 0; i **<** f**.**size(); **++**i){

int **&**ans **=** f[i] **=** 1;

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

**if** (a[j] **<** a[i]) ans **=** max(ans, f[j]**+**1);

}

printf("i=%d, a=%d, f=%d\n", i, a[i], f[i]);

}

cout **<<** **\***max\_element(f**.**begin(), f**.**end());

}

**Kết quả chạy**

Ảnh có chứa văn bản

Mô tả được tạo tự động

**1.7 Thực hành sửa lỗi thực thi và lỗi logic**

**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

In [ ]:

*#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;

}

**Phần 2. Kiểm thử**

Test đề bài thường là test đơn giản, không thể kiểm tra hết các ngõ ngách logic trong chương trình. Do đó chúng ta cần sinh thêm các test khác. Các loại test nên sinh bao gồm: test ngẫu nhiên, test đặc biệt (để kiểm tra các trường hợp đặc biệt), test lớn (để kiểm tra giới hạn và thời gian chạy)

Trong khi test đặc biệt có thể sinh bằng tay, thường nhỏ và phụ thuộc vào kinh nghiệm của lập trình viên, thì test ngẫu nhiên cần phải sinh nhiều và lớn tương đối. Do đó cần thiết phải viết chương trình sinh test và kiểm thử

**2.1 Hướng dẫn**

Đoạn code sau đây sẽ sinh input cho bài toán dãy con tăng dài nhất

In [ ]:

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

using namespace std;

int random(int L, int H){

**return** (rand()**<<**16**|**rand()) **%** (H**-**L**+**1) **+** L;

}

int main(int argc, char**\*\*** argv){

**if** (argc **>** 1) srand(atoi(argv[1]));

**else** srand(time(NULL));

freopen("LIS.inp", "w", stdout);

int n **=** random(1, 10), amax **=** 1e9;

cout **<<** n **<<** endl;

**for** (int i **=** 1; i **<=** n; **++**i) cout **<<** random(**-**amax, amax) **<<** " ";

}

Sau khi viết chương trình sinh test, có thể chạy bằng tay. Trong trường hợp output rất lớn hoặc khó kiểm tra, chúng ta nên viết một solution khác (đơn giản, chạy chậm nhưng chắc chắn đúng, thường gọi là thuật toán trực tiếp).

Nếu chỉ test cỡ chục test trở lại, với 3 chương trình trên (sol1, sol2, gentest) ta có thể chạy thủ công. Tuy nhiên có thể viết một file Shell script để làm điều này và lặp lại với hàng trăm test. Script này chỉ cần viết 1 lần và dùng được cho tất cả các bài mà không phải chỉnh sửa gì

In [ ]:

@echo off

**for** **/**l **%%i** in (1, 1, 100) do (

gentest.exe %%i

sol1.exe

sol2.exe

fc sol1.out sol2.out

if errorlevel 1 goto :eof

echo Test %%i correct

)

In [ ]:

**//**sol1 và sol 2

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

using namespace std;

int main(int argc, char**\*\*** argv){

freopen("LIS.inp", "r", stdin);

freopen("sol1.out", "w", stdout);

int n;

cin **>>** n;

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

**for** (int **&**x : a) cin **>>** x;

int ans **=** 0;

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

int pre **=** **-**2e9, L **=** 0;

**for** (int j **=** i; j **<** a**.**size(); **++**j){

**if** (a[j] **>** pre){

**++**L;

pre **=** a[j];

}

}

ans **=** max(ans, L);

}

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

}

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

using namespace std;

int main(int argc, char**\*\*** argv){

freopen("LIS.inp", "r", stdin);

freopen("sol2.out", "w", stdout);

int n;

cin **>>** n;

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

**for** (int **&**x : a) cin **>>** x;

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

**for** (int i **=** 0; i **<** f**.**size(); **++**i){

int **&**ans **=** f[i] **=** 1;

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

**if** (a[j] **<** a[i]) ans **=** max(ans, f[j]**+**1);

}

}

cout **<<** **\***max\_element(f**.**begin(), f**.**end());

}

**Kết quả kiểm thử**

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ta cũng có thể viết chương trình C++ thay cho shell script, và cũng dùng chung cho nhiều bài được

In [ ]:

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

using namespace std;

int main(){

**for**(int iTest **=** 1; iTest **<=** 100; iTest**++**){

char tmp[100];

sprintf(tmp, "%d", iTest);

string seed(tmp);

system((string("gentest.exe ") **+** seed)**.**c\_str());

system(string("sol1.exe")**.**c\_str());

system(string("sol2.exe")**.**c\_str());

**if**(system(string("fc sol1.out sol2.out")**.**c\_str()) **!=** 0){

cout **<<** "Test " **<<** iTest **<<** ": WRONG!\n";

**return** 0;

}

cout **<<** "Test " **<<** iTest **<<** ": CORRECT!\n";

}

}

**Kết quả kiểm thử**

Ảnh có chứa văn bản

Mô tả được tạo tự động

Trên thực tế, các project khi làm việc đã có sẵn bộ test. Lúc này chúng ta chỉ cần chạy chương trình và so sánh kết quả, không cần phải sinh test (bằng gentest và sol2) nữa. File script có thể sửa lại như sau

In [ ]:

@echo off

**for** **/**l **%%i** in (1, 1, 100) do (

copy %%i.inp LIS.inp

LIS.exe

fc LIS.out %%i.out

if errorlevel 1 goto :eof

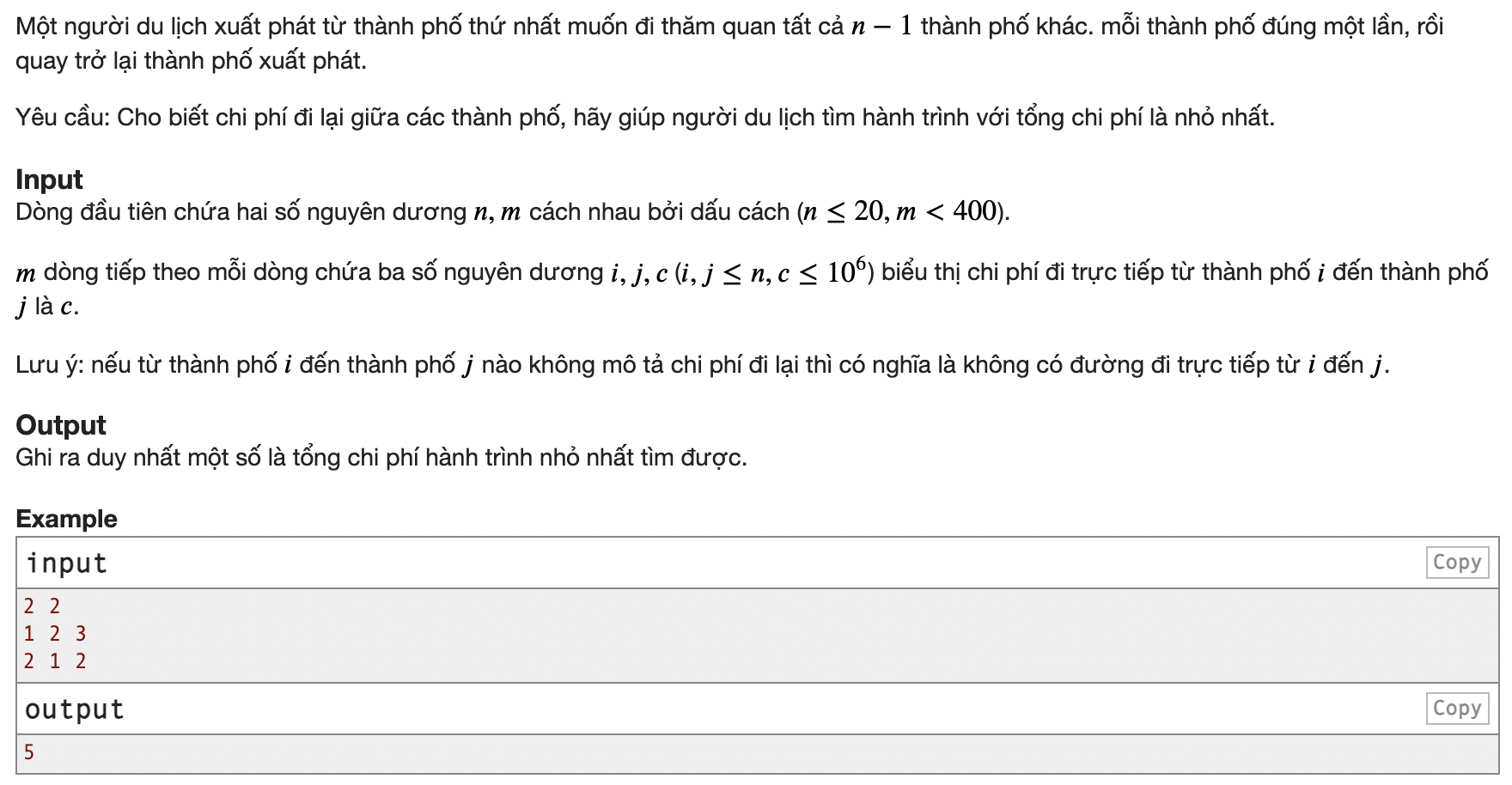
echo Test %%i correct

)

**2.2 Thực hành**

Viết chương trình sinh test và kiểm thử cho các bài toán sau

**Bài tập 4: Bài toán người du lịch**

Đề bài:  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.

In [ ]:

*#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;

}

In [ ]:

*#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;

}

**Phần 3. Tinh chỉnh mã nguồn**

**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.

In [ ]:

*#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";

}

**Bài tập 6: Tổng kết**

Một lớp có nn sinh viên. Sinh viên thứ ii có điểm tổng kết là aiai 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<4a<4: F

4≤a<5.54≤a<5.5: D

5.5≤a<75.5≤a<7: C

7≤a<8.57≤a<8.5: B

8.5≤a8.5≤a: A

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

In [ ]:

*#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;

}

**Bài tập 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 nn thôn, thôn thứ ii có kiki 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 ii giàu, họ sẽ đi từng nhà trong số kiki nhà này và xin ai,jai,j tiền của nhà thứ jj. Nếu thôn ii nghèo, họ sẽ đi từng nhà trong số kiki nhà này và phát ai,jai,j tiền cho nhà thứ jj. 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.

In [ ]:

*#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;

}

**Phần 4. Bài tập về nhà**

**Bài tập 8: Cắt hình chữ nhật**

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

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

In [ ]:

*#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;

}

**Bài tập 9: Xây tháp**

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

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

In [ ]:

*#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; n **<=** n; i**++**){

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;

}

}

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 **<=** 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(){

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;

}