Contents

[Bài thực hành 3: Đệ quy và khử đệ quy để giải quyết một số bài toán 2](#_Toc166887377)

[Phần 1. Thực hành về đệ quy 2](#_Toc166887378)

[1.1 Đệ quy - quay lui 2](#_Toc166887379)

[Bài tập 1: Tính dãy Lucas 2](#_Toc166887380)

[Bài tập 2: Quân mã đi tuần 4](#_Toc166887382)

[1.2 Kỹ thuật nhánh cận 8](#_Toc166887384)

[Bài tập 3: Bài toán người du lịch 8](#_Toc166887385)

[1.3 Đệ quy có nhớ 10](#_Toc166887387)

[Bài tập 4: LIS 10](#_Toc166887388)

[Phần 2. Khử đệ quy 12](#_Toc166887390)

[Bài tập 5: Tính tổ hợp 13](#_Toc166887391)

[Bài tập 6: Tìm ước chung lớn nhất 15](#_Toc166887393)

[Bài tập 7: Liệt kê xâu nhị phân 16](#_Toc166887395)

[Bài tập 8: Cân đĩa 20](#_Toc166887397)

[Phần 3. Bài tập về nhà 22](#_Toc166887399)

[Bài tập 9: Lập lịch cho y tá 22](#_Toc166887401)

[Bài tập 10: Khoảng cách Hamming 25](#_Toc166887403)

[Bài tập 11: Lịch trình chụp ảnh 28](#_Toc166887405)

[Bài tập 12: Đếm đường đi 32](#_Toc166887407)

# Bài thực hành 3: Đệ quy và khử đệ quy để giải quyết một số bài toán

# Phần 1. Thực hành về đệ quy

## 1.1 Đệ quy - quay lui

### Bài tập 1: Tính dãy Lucas

**Dãy Lucas được định nghĩa bởi Ln=Ln−1+Ln−2 và bắt đầu bởi L0=2, L1=1. Viết**

### hàm tính số Lucas thứ nn.

//Nhữ Ngọc Minh -20225046

int lucas(int n)

{

if (n==0) return 2;

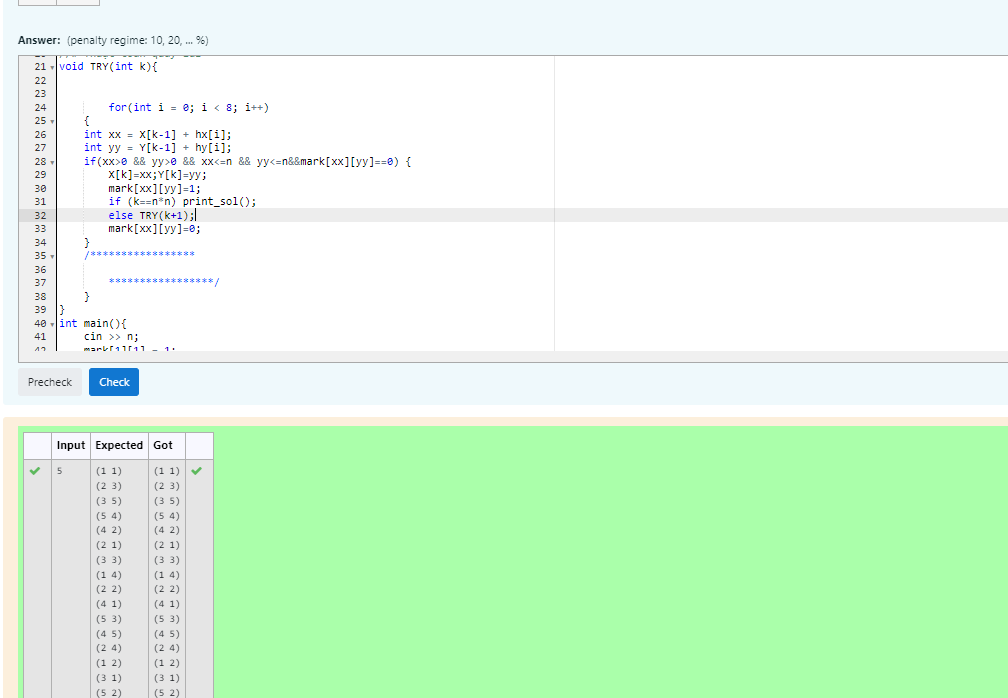
else if (n==1) return 1;

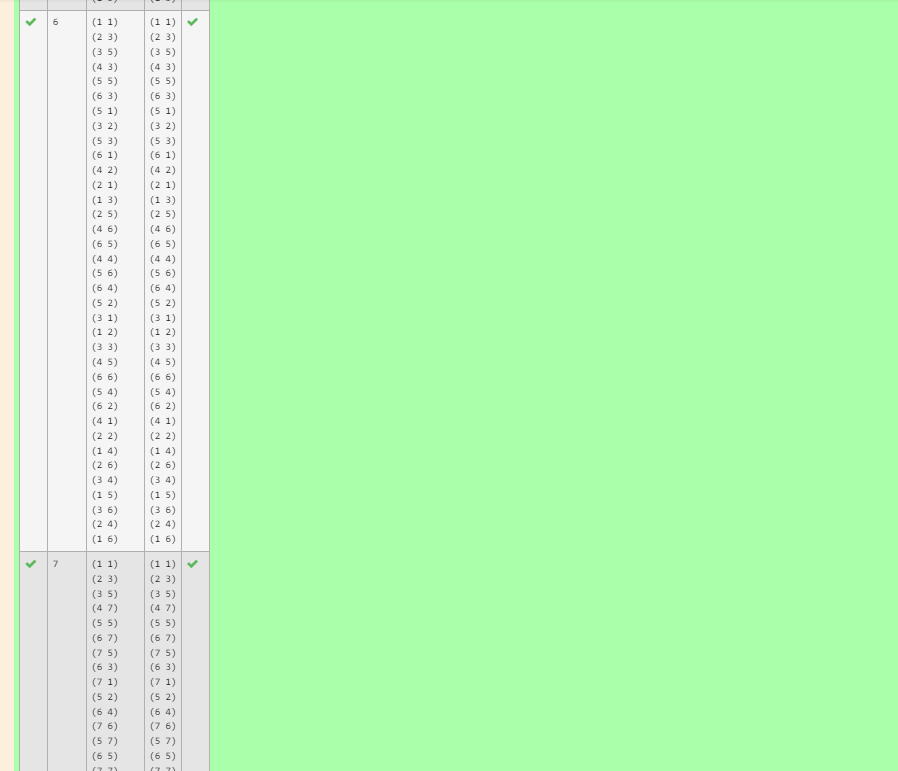
else return lucas(n-1)+lucas(n-2);

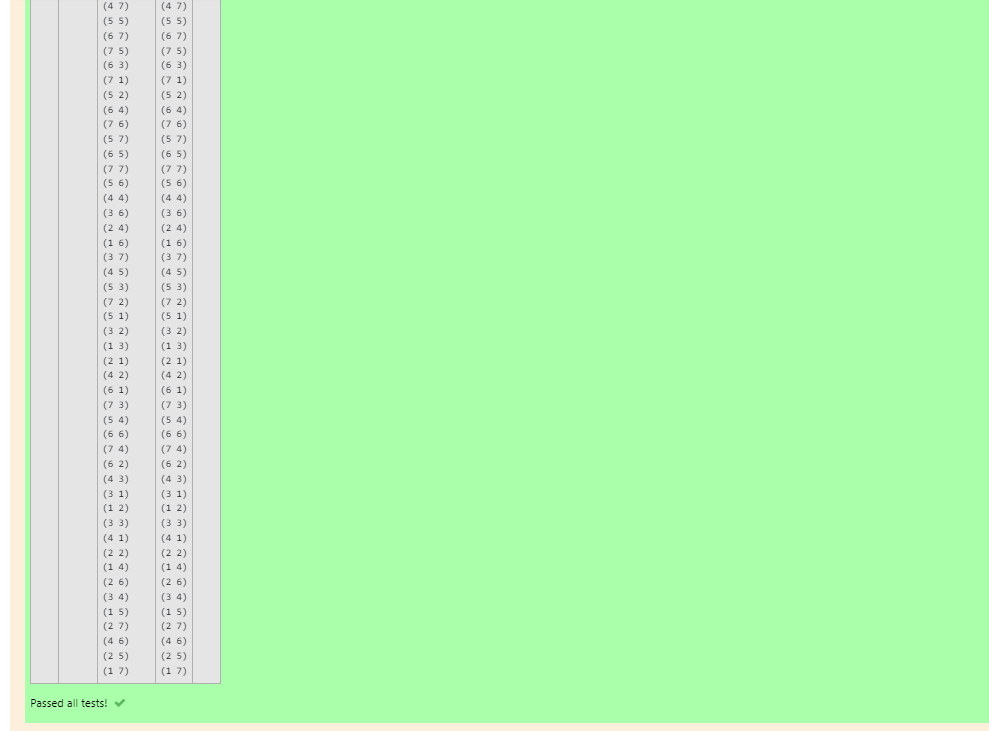
}

### Bài tập 2: Quân mã đi tuần

### Trên bàn cờ vua kích thước n×n có một quân mã đang ở ô (1, 1). Hãy đưa ra một dãy các di chuyển của mã sao cho mỗi ô trên bàn cờ đều được đi qua đúng 1 lần (ô (1, 1) được xem là đã đi qua)







//Nhữ Ngọc Minh- 20225046

#include <iostream>

using namespace std;

int n;

int X[100], Y[100];

//# Lưu tọa độ các bước di chuyển của quân mã

int mark[100][100]; //# Đánh dấu vị trí các ô mà quân mã đã di chuyển qua

//# Mảng hx, hy mô tả 8 vị trí quân mã có thể di chuyển kể từ vị trí hiện tại

const int hx[] = {1, 1, 2, 2, -1, -1, -2, -2}; const int hy[] = {2, -2, 1, -1, 2, -2, 1, -1};

//# In ra dãy các di chuyển tìm được

void print\_sol(){

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

printf("(%d %d)\n", X[j], Y[j]);

exit(0);

}

//# Thuật toán quay lui

void TRY(int k){

for(int i = 0; i < 8; i++)

{

int xx = X[k-1] + hx[i];

int yy = Y[k-1] + hy[i];

if(xx>0 && yy>0 && xx<=n && yy<=n&&mark[xx][yy]==0) {

X[k]=xx;Y[k]=yy;

mark[xx][yy]=1;

if (k==n\*n) print\_sol();

else TRY(k+1);

mark[xx][yy]=0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

}

int main(){

cin >> n;

mark[1][1] = 1;

X[1] = Y[1] = 1;

TRY(2);

return 0; }

## 1.2 Kỹ thuật nhánh cận

### Bài tập 3: 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 ii sang thành phố jj là ci,jci,j. Hãy tìm tổng chi phí nhỏ nhất có thể

**Dữ liệu vào:**

Dòng 1: Chứa số nguyên nn (1≤n≤161≤n≤16) nn dòng tiếp theo: Chứa ma trận cc (0≤ci,j≤10000000≤ci,j≤1000000)

**Kết quả:**

Ghi tổng chi phí nhỏ nhất có thể

**Ví dụ:**

**Dữ liệu mẫu:** 4

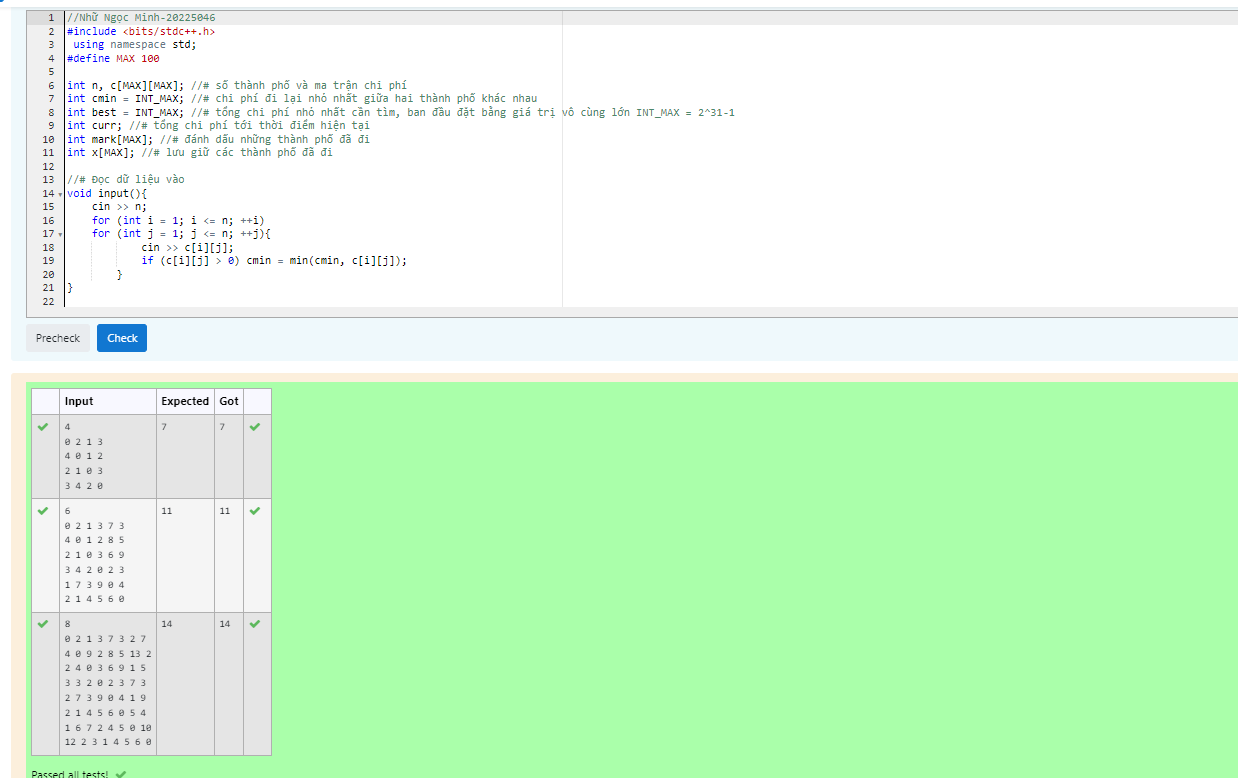
0 2 1 3

4 0 1 2

1. 1 0 3
2. 4 2 0

**Kết quả mẫu:**

7



//Nhữ Ngọc Minh-20225046

#include <bits/stdc++.h>

using namespace std;

#define MAX 100

int n, c[MAX][MAX]; //# số thành phố và ma trận chi phí

int cmin = INT\_MAX; //# chi phí đi lại nhỏ nhất giữa hai thành phố khác nhau

int best = INT\_MAX; //# tổng chi phí nhỏ nhất cần tìm, ban đầu đặt bằng giá trị vô cùng lớn INT\_MAX = 2^31-1

int curr; //# tổng chi phí tới thời điểm hiện tại

int mark[MAX]; //# đánh dấu những thành phố đã đi

int x[MAX]; //# lưu giữ các thành phố đã đi

//# Đọc dữ liệu vào

void input(){

cin >> n;

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

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

cin >> c[i][j];

if (c[i][j] > 0) cmin = min(cmin, c[i][j]);

}

}

//# Thuật toán quay lui

void TRY(int k)

{

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

{

if (!mark[i])

{

curr+=c[x[k-1]][i];

mark[i]=1;

x[k]=i;

if(k==n)

{

if (curr+c[x[k]][1]<best) best=curr+c[x[k]][1];

}

else TRY(k+1);

curr-=c[x[k-1]][i];

mark[i]=0;

}

}

}

int main() {

input();

x[1] = 1;

TRY(2);

cout << best;

return 0;

}

## 1.3 Đệ quy có nhớ

### Bài tập 4: LIS

### Cho dãy aa có nn phần tử. Một dãy con của aa là dãy thu được bằng cách xóa đi một số phần tử của aa và giữ nguyên thứ tự các phần tử còn lại (có thể không xóa phần tử nào). Hãy tìm dãy con tăng dài nhất của aa

**Dữ liệu vào:**

Dòng 1: Chứa số nguyên nn (1≤n≤10001≤n≤1000)

Dòng 2: Chứa nn số nguyên a1a1 a2a2 …… anan (|ai|≤109|ai|≤109)

**Kết quả:**

Dòng đầu tiên chứa độ dài dãy con tăng dài nhất

Dòng thứ 2 chứa chỉ số các phần tử được chọn vào dãy con đó Nếu có nhiều dãy con tăng dài nhất, in ra dãy bất kỳ trong số đó

**Ví dụ:**

**Dữ liệu mẫu:**

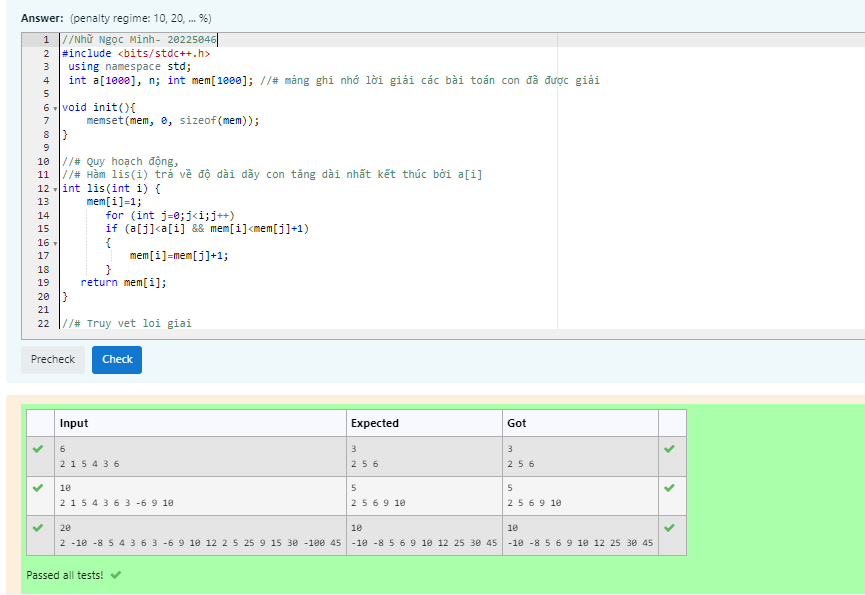
6

2 1 5 4 3 6

**Kết quả mẫu:**

3

1. 5 6



//Nhữ Ngọc Minh- 20225046

#include <bits/stdc++.h>

using namespace std;

int a[1000], n; int mem[1000]; //# mảng ghi nhớ lời giải các bài toán con đã được giải

void init(){

memset(mem, 0, sizeof(mem));

}

//# Quy hoạch động,

//# Hàm lis(i) trả về độ dài dãy con tăng dài nhất kết thúc bởi a[i]

int lis(int i) {

mem[i]=1;

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

if (a[j]<a[i] && mem[i]<mem[j]+1)

{

mem[i]=mem[j]+1;

}

return mem[i];

}

//# Truy vet loi giai

void trace(int i)

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

{ if (a[j] < a[i] && mem[i] == 1 + mem[j])

{ trace(j);

break;

}

}

cout << a[i] << " ";

}

int main(){

init();

cin >> n;

for(int i = 0; i < n; i++) cin >> a[i];

int res = 1, pos = 0; mem[0]=1;

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

{ if (res < lis(i)){

res = lis(i);

pos = i;

}

}

cout << res << endl;

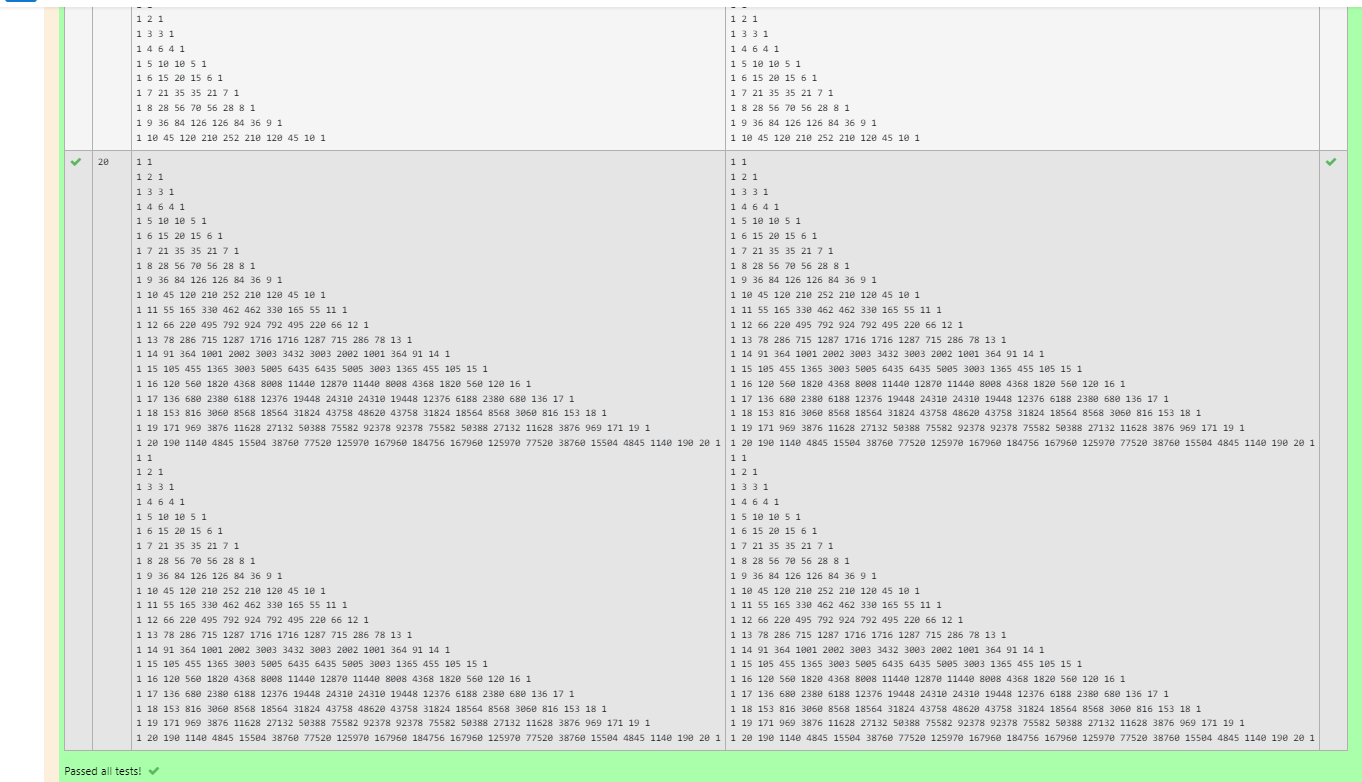
trace(pos); return 0; }

# Phần 2. Khử đệ quy

Hãy giải các bài toán sau đây bằng phương pháp khử đệ quy

### Bài tập 5: Tính tổ hợp

### Tính CknCnk



//Nhữ Ngọc Minh-20225046

#include <iostream>

using namespace std;

#define max 1000

int luu[max][max];

int binom(int n, int k) {

if (k > n) return 0;

if (k == 0) return 1;

return binom(n-1, k) + binom(n-1, k-1);

}

int binom2(int n, int k){

int tmp1,tmp2;

if(k>n) return 0;

if (k==0) return 1;

if (luu[n-1][k]) tmp1=luu[n-1][k];

else tmp1=binom2(n-1,k);

if (luu[n-1][k-1]) tmp2=luu[n-1][k-1];

else tmp2=binom2(n-1,k-1);

return tmp1+tmp2;

}

int main() {

int m;

cin >> m;

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

for (int k = 0; k <= n; ++k)

printf("%d ", binom(n, k));

printf("\n");

}

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

for (int k = 0; k <= n; ++k)

printf("%d ", binom2(n, k));

printf("\n");

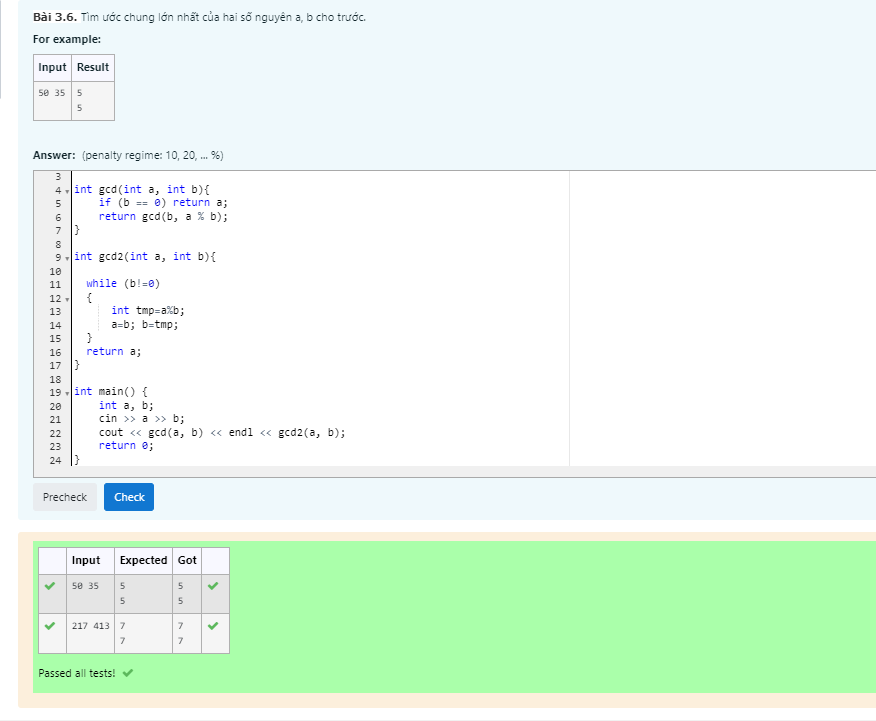
}

return 0;

}

### Bài tập 6: Tìm ước chung lớn nhất

### Tính ước chung lớn nhất của hai số cho trước



//Nhữ Ngọc Minh- 20225046

#include <iostream>

using namespace std;

int gcd(int a, int b){

if (b == 0) return a;

return gcd(b, a % b);

}

int gcd2(int a, int b){

while (b!=0)

{

int tmp=a%b;

a=b; b=tmp;

}

return a;

}

int main() {

int a, b;

cin >> a >> b;

cout << gcd(a, b) << endl << gcd2(a, b);

return 0;

}

### Bài tập 7: Liệt kê xâu nhị phân

### Sử dụng phương pháp khử đệ quy bằng stack, hãy liệt kê các xâu nhị phân độ dài nn không có kk bit 1 nào liên tiếp

**Dữ liệu vào:**

Một dòng duy nhất chứa hai số nguyên nn kk (1≤k≤n≤201≤k≤n≤20)

**Kết quả:**

Với mỗi xâu tìm được, in ra nn ký tự trên một dòng, các ký tự cách nhau bởi dấu cách. Các xâu cần được liệt kê theo thứ tự từ điển

**Ví dụ:**

**Dữ liệu mẫu:** 4 2

**Kết quả mẫu:**

0 0 0 0

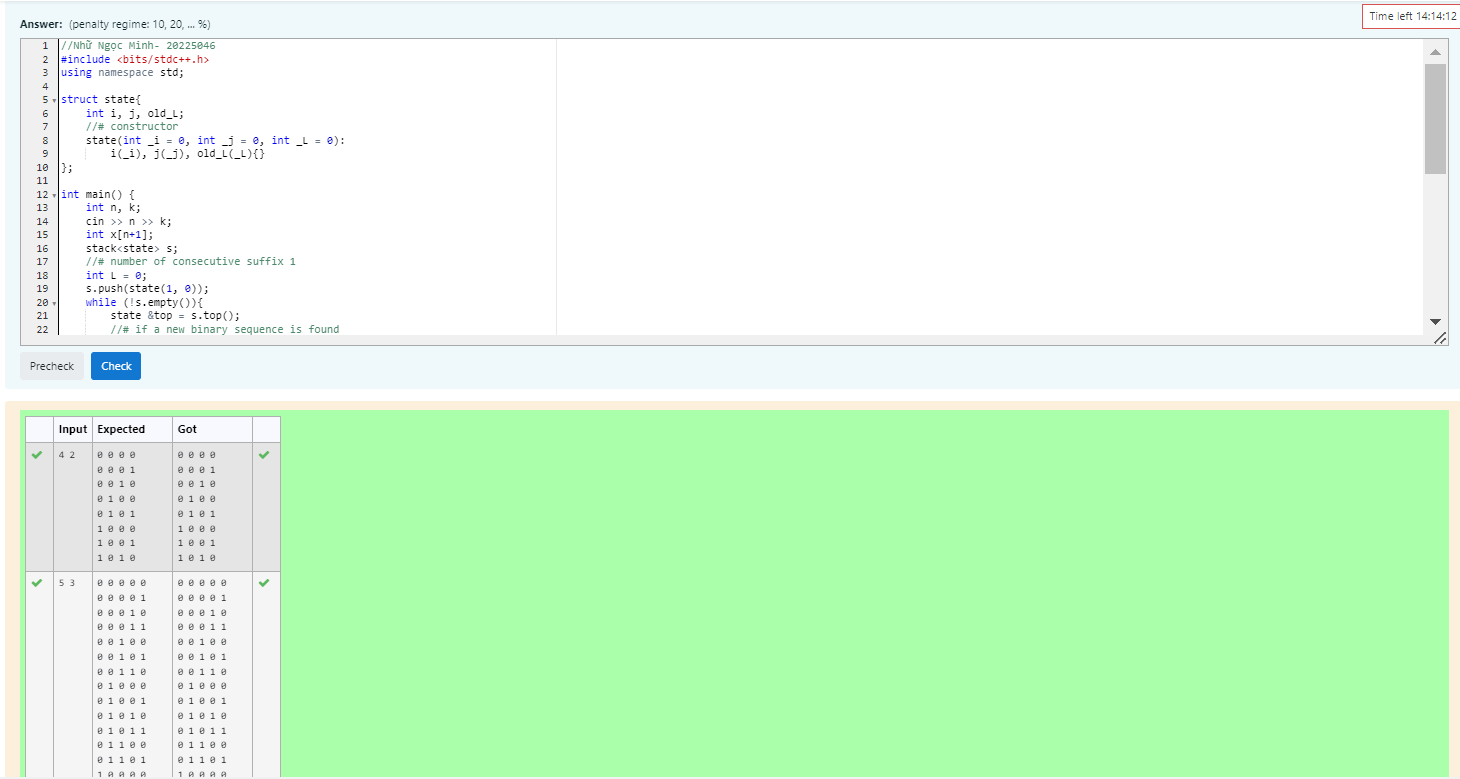
0 0 0 1

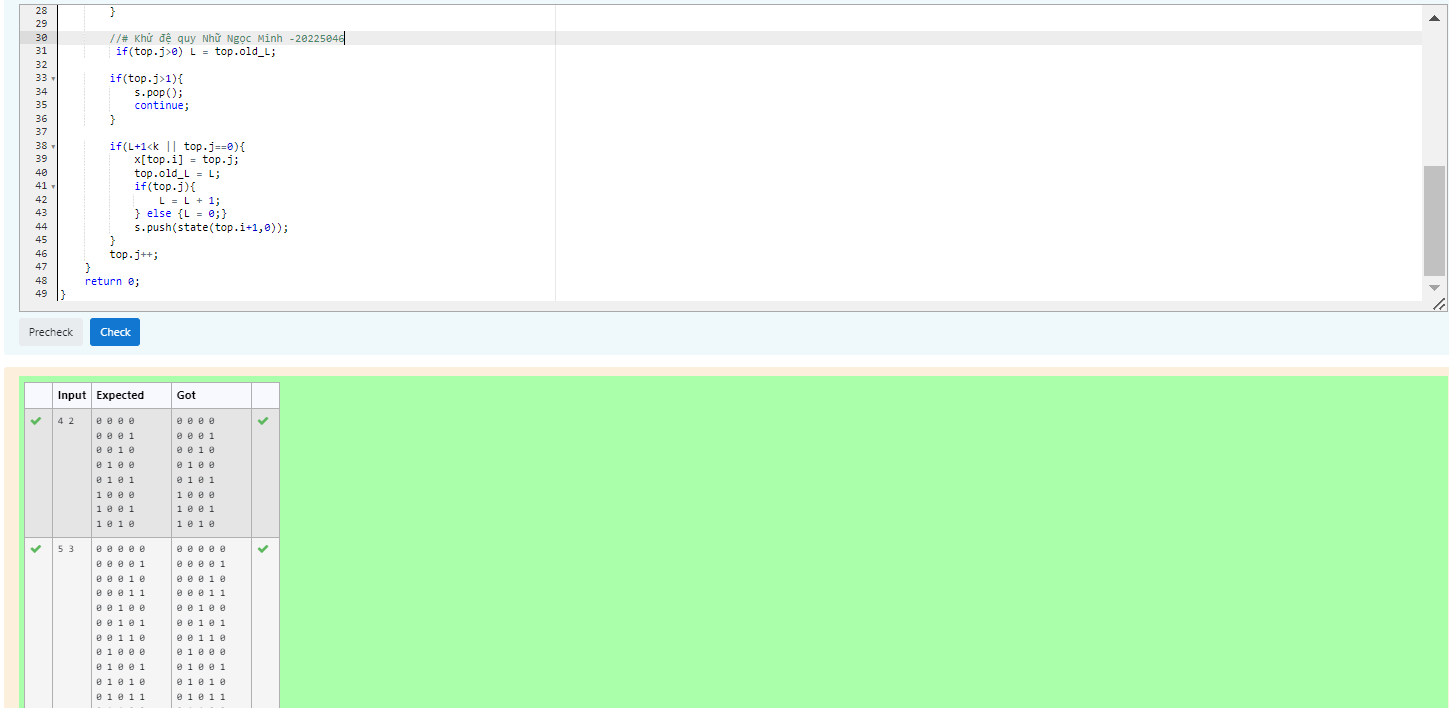
0 0 1 0

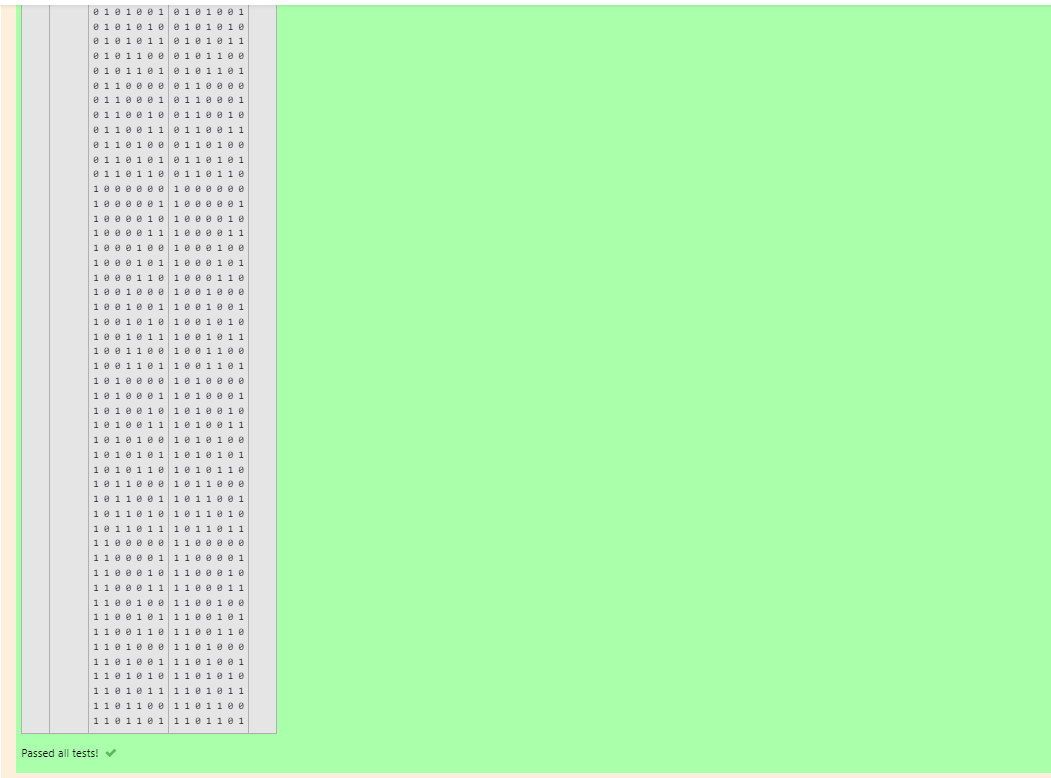
0 1 0 0

1. 1 0 1
2. 0 0 0

1 0 0 1







//Nhữ Ngọc Minh- 20225046

#include <bits/stdc++.h>

using namespace std;

struct state{

int i, j, old\_L;

//# constructor

state(int \_i = 0, int \_j = 0, int \_L = 0):

i(\_i), j(\_j), old\_L(\_L){}

};

int main() {

int n, k;

cin >> n >> k;

int x[n+1];

stack<state> s;

//# number of consecutive suffix 1

int L = 0;

s.push(state(1, 0));

while (!s.empty()){

state &top = s.top();

//# if a new binary sequence is found

if (top.i > n){

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

cout << x[i] << " \n"[i == n];

s.pop();

continue;

}

//# Khử đệ quy Nhữ Ngọc Minh -20225046

if(top.j>0) L = top.old\_L;

if(top.j>1){

s.pop();

continue;

}

if(L+1<k || top.j==0){

x[top.i] = top.j;

top.old\_L = L;

if(top.j){

L = L + 1;

} else {L = 0;}

s.push(state(top.i+1,0));

}

top.j++;

}

return 0;

}

### Bài tập 8: Cân đĩa

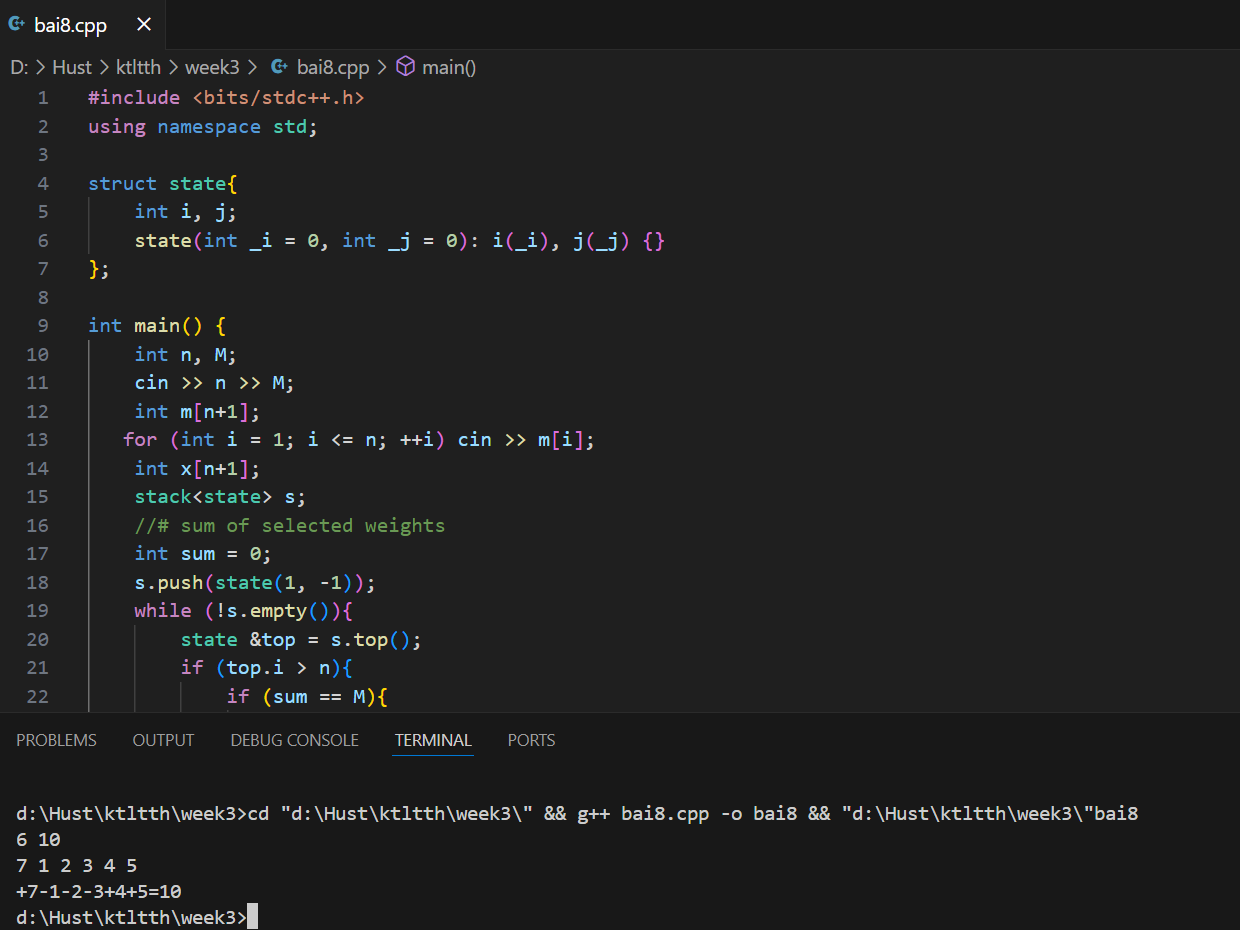
### Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng MM như người ta nói hay không. Có một cân thăng bằng và nn quả cân. Quả thứ ii nặng mimi. Hãy chỉ ra một cách cân thỏa mãn. Quy cách in ra đã được tích hợp trong mã nguồn dưới.

**Dữ liệu mẫu:**

1. 10
2. 1 2 3 4 5

**Kết quả mẫu:**

-1+2+3+4+5=10



#include <bits/stdc++.h>

using namespace std;

struct state{

    int i, j;

    state(int \_i = 0, int \_j = 0): i(\_i), j(\_j) {}

};

int main() {

    int n, M;

    cin >> n >> M;

    int m[n+1];

   for (int i = 1; i <= n; ++i) cin >> m[i];

    int x[n+1];

    stack<state> s;

    //# sum of selected weights

    int sum = 0;

    s.push(state(1, -1));

    while (!s.empty()){

        state &top = s.top();

        if (top.i > n){

            if (sum == M){

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

                    if (x[i] == -1) cout << '-' << m[i];

                    if (x[i] == 1) cout << '+' << m[i];

                }

                cout << "=" << M;

                exit(0);

            }}

            x[top.i]=top.j;

            if (top.j==-1||top.j==1) {

                if (top.j==-1)

                sum-=(m[top.i]);

                else if (top.j==1)

                sum+=2\*(m[top.i]);

                if(top.i<=n) s.push(state(top.i+1,-1));

            }

            else if (top.j>1) {

                sum-=m[top.i];

                s.pop();

            continue;}

          top.j+=2;

    }

    cout << -1;

    return 0;

}

# Phần 3. Bài tập về nhà

# Sinh viên tự làm các bài tập sau:

### Bài tập 9: Lập lịch cho y tá

# Một y tá cần lập lịch làm việc trong NN ngày, mỗi ngày chỉ có thể là làm việc hay nghỉ ngơi. Một lịch làm việc là tốt nếu không có hai ngày nghỉ nào liên tiếp và mọi chuỗi ngày tối đại làm việc liên tiếp đều có số ngày thuộc đoạn [K1,K2][K1,K2]. Hãy liệt kê tất cả các cách lập lịch tốt, với mỗi lịch in ra trên một dòng một xâu nhị phân độ dài nn với bit 0/1 tương ứng là nghỉ/làm việc. Các xâu phải được in ra theo thứ tự từ điển

**Dữ liệu vào:**

Ghi 3 số nguyên N,K1,K2N,K1,K2 (N≤200,K1<K2≤70N≤200,K1<K2≤70)

**Kết quả:**

Ghi danh sách các lịch tìm được theo thứ tự từ điển

**Ví dụ:**

**Dữ liệu mẫu:**

6 2 3

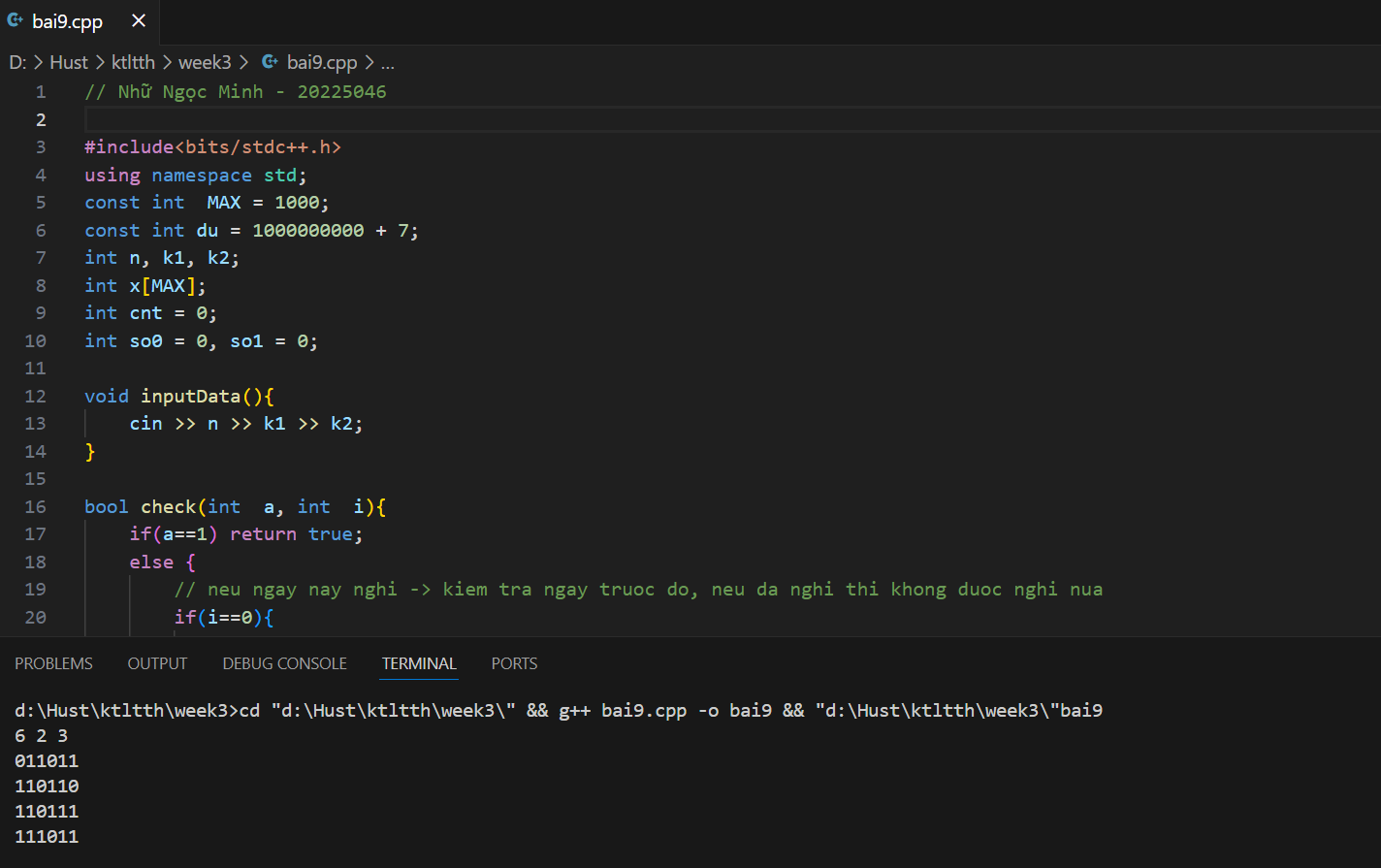
**Kết quả mẫu:**

011011

110110

110111

111011



// Nhữ Ngọc Minh - 20225046

#include<bits/stdc++.h>

using namespace std;

const int  MAX = 1000;

const int du = 1000000000 + 7;

int n, k1, k2;

int x[MAX];

int cnt = 0;

int so0 = 0, so1 = 0;

void inputData(){

    cin >> n >> k1 >> k2;

}

bool check(int  a, int  i){

    if(a==1) return true;

    else {

        // neu ngay nay nghi -> kiem tra ngay truoc do, neu da nghi thi khong duoc nghi nua

        if(i==0){

            if(x[a-1] == 0) return false;

            else { // x[a-1] == 1

                if(so1<k1) return false;

            }

        } else { // i == 1

            if(x[a-1] == 0){

                if(n-a+1 < k1) return false;

            } else {

                if(so1>=k2) return false;

            }

        }

        return true;

    }

}

void solution(){

    for(int i=1; i<=n; i++)

        cout << x[i];

    cout << endl;

    cnt++;

}

void TRY(int  a){

    for(int  i=0; i<=1; i++){

        if(check(a,i)){

            x[a] = i; // ngay thu a lam viec hoac khong lam viec

            int pre = so1;

            if(i == 1){

                if(x[a-1] == 1) so1++;

                else so1 = 1;

            } else {

                so1 = 0;

            }

            if(a==n) solution();

            else TRY(a+1);

            so1 = pre;

        }

    }

}

int main(){

    inputData();

    TRY(1);

    return 0;

}

### Bài tập 10: Khoảng cách Hamming

# Khoảng cách Hamming giữa hai xâu cùng độ dài là số vị trí mà ký tự tại vị trí đó là khác nhau trên hai xâu. Cho SS là xâu gồm nn ký tự 0. Hãy liệt kê tất cả các xâu nhị phân độ dài nn, có khoảng cách Hamming với SS bằng HH. Các xâu phải được liệt kê theo thứ tự từ điển

**Dữ liệu vào:**

Dòng đầu chứa TT là số testcase

TT dòng tiếp theo, mỗi dòng mô tả một testcase, ghi NN và HH (1≤H≤N≤161≤H≤N≤16)

**Kết quả:**

Với mỗi testcase, in ra danh sách các xâu thỏa mãn. In ra một dòng trống giữa hai testcase

**Ví dụ:**

**Dữ liệu mẫu:**

2

4 2

1 0

**Kết quả mẫu:**

0011

0101

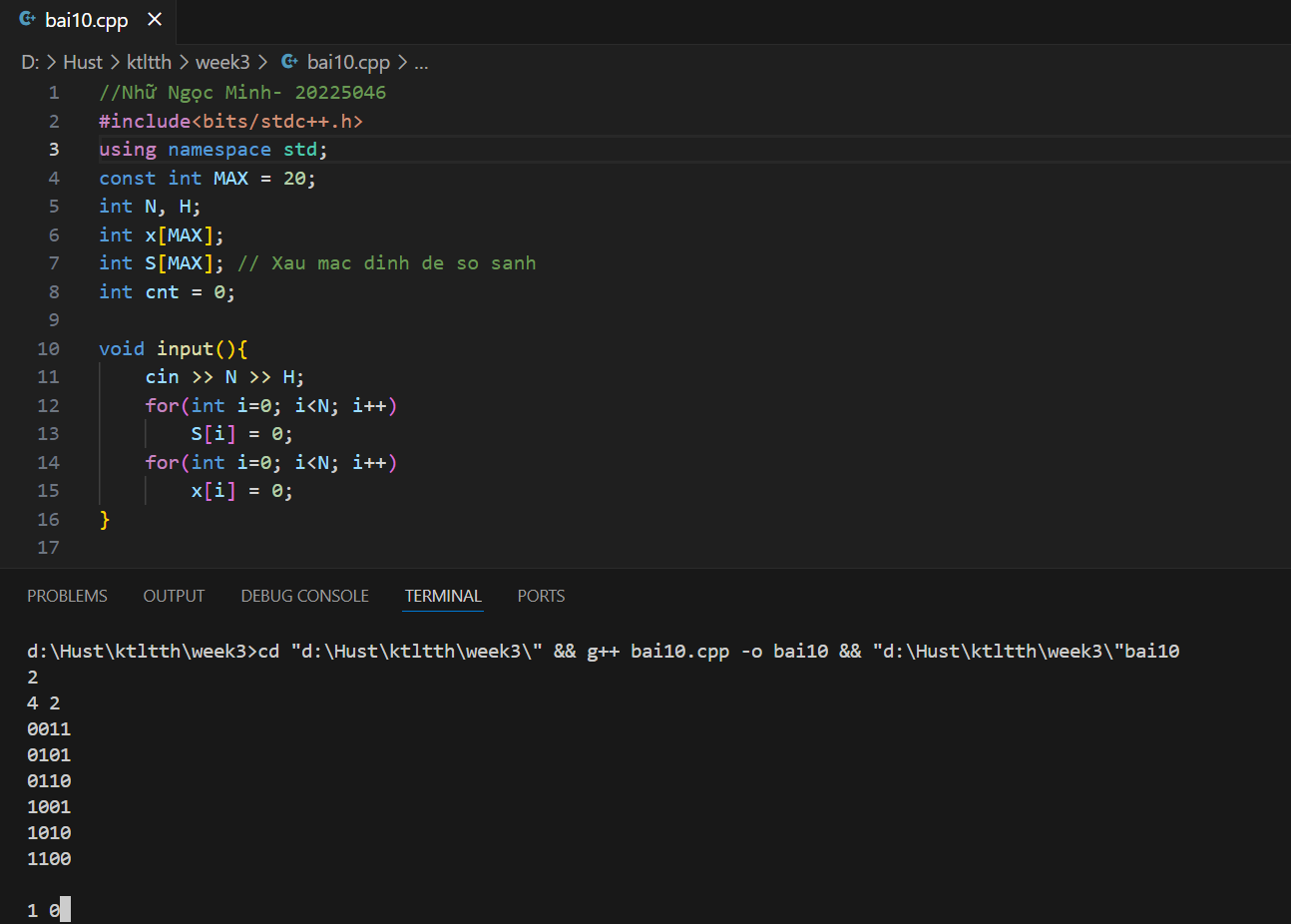
0110

1001

1010

1100

0



//Nhữ Ngọc Minh- 20225046

#include<bits/stdc++.h>

using namespace std;

const int MAX = 20;

int N, H;

int x[MAX];

int S[MAX]; // Xau mac dinh de so sanh

int cnt = 0;

void input(){

    cin >> N >> H;

    for(int i=0; i<N; i++)

        S[i] = 0;

    for(int i=0; i<N; i++)

        x[i] = 0;

}

// Kiem tra 2 co khoang cach Hamming la bnh

int checkHamming(int str1[], int str2[]){

    int lens = N;

    int cnt = 0;

    for(int i=0; i<lens; i++){

        if(str1[i] != str2[i]) cnt++;

    }

    return cnt;

}

bool check(int a, int i){

    return true;

}

void solution(){

    if(checkHamming(x,S) == H){

        for(int i=0; i<N; i++)

            cout << x[i];

        cout << endl;

    }

}

void TRY(int a){

    for(int i=0; i<=1; i++){

        x[a] = i;

        if(a == N-1) solution();

        else

            TRY(a+1);

    }

}

int main(){

    int T;

    cin >> T;

    while(T > 0){

        input();

        TRY(0);

        T--;

    }

}

### Bài tập 11: Lịch trình chụp ảnh

# Superior là một hòn đảo tuyệt đẹp với nn địa điểm chụp ảnh và các đường một chiều nối các điểm chụp ảnh với nhau. Đoàn khách tham quan có rr người với sở thích chụp ảnh khác nhau. Theo đó, mỗi người sẽ đưa ra danh sách các địa điểm mà họ muốn chụp. Bạn cần giúp mỗi người trong đoàn lập lịch di chuyển sao cho đi qua các điểm họ yêu cầu đúng một lần, không đi qua điểm nào khác, bắt đầu tại điểm đầu tiên và kết thúc tại điểm cuối cùng trong danh sách mà họ đưa ra, và có tổng khoảng cách đi lại là nhỏ nhất.

**Dữ liệu vào:**

Dòng đầu chứa nn và rr

Tiếp theo là ma trận n×nn×n mô tả chi phí đi lại giữa các địa điểm. Chi phí bằng 0 có nghĩa là không thể đi lại giữa hai địa điểm đó.

rr dòng tiếp theo chứa danh sách các địa điểm mà người rr đưa ra. Lưu ý là hành mỗi hành trình cần phải bắt đầu và kết thúc bởi hai đỉnh đầu và cuối của danh sách, còn các địa điểm còn lại có thể thăm theo bất kỳ thứ tự nào

**Kết quả:**

Gồm rr dòng ghi chi phí đi lại ít nhất của rr người theo thứ tự đầu vào

**Ví dụ:**

**Dữ liệu mẫu:**

6 3

1. 1 2 0 1 1
2. 0 1 1 1 0

0 2 0 1 3 0

4 3 1 0 0 0

1. 0 1 1 0 0
2. 0 0 0 0 0

1 3 5

6 3 2 5

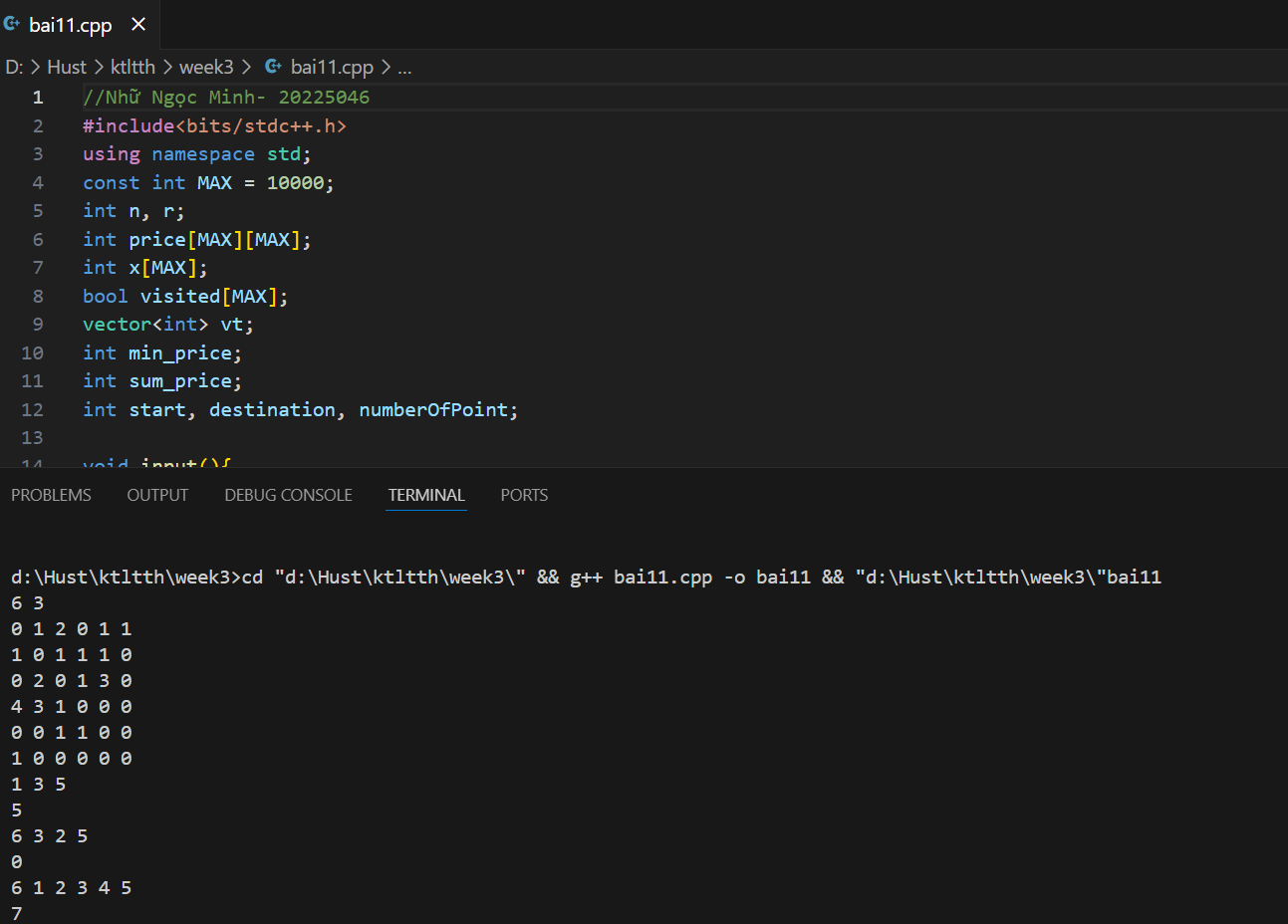
6 1 2 3 4 5

**Kết quả mẫu:**

5

0

7



//Nhữ Ngọc Minh- 20225046

#include<bits/stdc++.h>

using namespace std;

const int MAX = 10000;

int n, r;

int price[MAX][MAX];

int x[MAX];

bool visited[MAX];

vector<int> vt;

int min\_price;

int sum\_price;

int start, destination, numberOfPoint;

void input(){

    cin >> n >> r;

    for(int i=0; i<n; i++)

        for(int j=0; j<n; j++){

            cin >> price[i][j];

        }

}

bool check(int a, int i){

    if(visited[vt[i]]) return false;

    if(price[x[a-1]][vt[i]] == 0) return false;

    return true;

}

void solution(){

    if(price[x[numberOfPoint-2]][destination] == 0) return;

    min\_price = min(min\_price, sum\_price + price[x[numberOfPoint-2]][destination]);

}

void TRY(int a){

    for(int i=1; i<numberOfPoint-1; i++){

        if(check(a, i)){

            visited[vt[i]] = true;

            sum\_price += price[x[a-1]][vt[i]];

            x[a] = vt[i];

            if(a == numberOfPoint-2) solution();

            else TRY(a+1);

            visited[vt[i]] = false;

            sum\_price -= price[x[a-1]][vt[i]];

        }

    }

}

int main(){

    string str;

    input(); getline(cin,str);

    while(r > 0){

        min\_price = INT\_MAX;

        sum\_price = 0;

        getline(cin, str);

        // Tach str thanh cac so va ghi vao vector vt

        /\*

        int pre = 0;

        for(int i=0; i<str.length(); i++){

            if(str[i] == ' '){

                string tmp = str.substr(pre,i);

                pre = i + 1;

                stringstream convert(tmp);

                int tmp\_int = 0;

                convert >> tmp\_int;

                vt.push\_back(tmp\_int - 1);

            }

        }

        string tmp = str.substr(pre,str.length());

        stringstream convert(tmp);

        int tmp\_int = 0;

        convert >> tmp\_int;

        vt.push\_back(tmp\_int - 1);

        \*/

        while (!str.empty()){

            stringstream convert(str.substr(0, str.find(" ")));

            int tmp = 0;

            convert >> tmp;

            vt.push\_back(tmp - 1);

            if (str.find(" ") > str.size()){

                break;

            } else {

                str.erase(0, str.find(" ") + 1); // Update string

            }

        }

        // Bat dau khoi tao cac du lieu can thiet truoc khi quay lui

        start = vt[0]; // diem bat dau dau

        destination = vt[vt.size()-1]; // diem dich

        numberOfPoint = vt.size(); // so diem phai di qua

        x[0] = start; x[numberOfPoint-1] = destination;

        for(int i=0; i<n; i++)

            visited[i] = false;

        TRY(1);

        // In ra ket qua

        if(min\_price == INT\_MAX) cout << "0" << endl;

        else cout << min\_price << endl;

        // Xoa vector va chuyen sang khach tiep theo

        vt.erase(vt.begin(), vt.end());

        r--;

    }

}

### Bài tập 12: Đếm đường đi

**Cho đồ thị vô hướng GG, hãy đếm số đường đi đi qua kk cạnh và không đi qua đỉnh nào quá một lần.**

**Dữ liệu vào:**

Dòng 1: Chứa hai số nguyên nn và kk (1≤n≤301≤n≤30, 1≤k≤101≤k≤10) với nn là số đỉnh của GG. Các đỉnh sẽ được đánh số từ 1 đến nn

Dòng 2: Chứa số nguyên mm (1≤m≤601≤m≤60) là số cạnh của GG mm dòng tiếp theo: Mỗi dòng chưa hai số nguyên là một cạnh của GG

**Kết quả:**

Số lượng đường đi đơn độ dài kk

**Ví dụ:**

**Dữ liệu mẫu:**

4 3

5

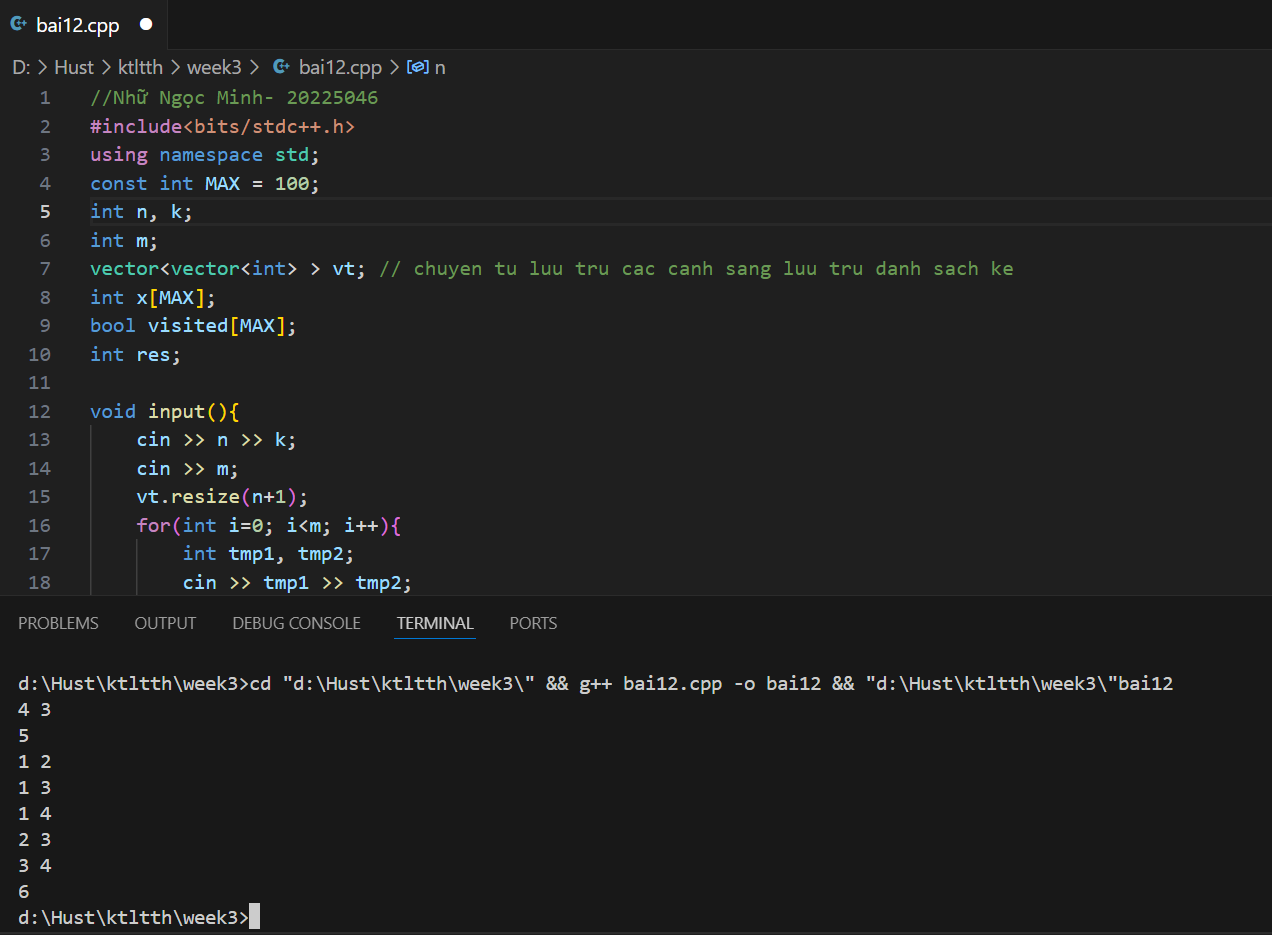
1 2

1 3

1. 4
2. 3
3. 4

**Kết quả mẫu:**

6



//Nhữ Ngọc Minh- 20225046

#include<bits/stdc++.h>

using namespace std;

const int MAX = 100;

int n, k;

int m;

vector<vector<int> > vt; // chuyen tu luu tru cac canh sang luu tru danh sach ke

int x[MAX];

bool visited[MAX];

int res;

void input(){

    cin >> n >> k;

    cin >> m;

    vt.resize(n+1);

    for(int i=0; i<m; i++){

        int tmp1, tmp2;

        cin >> tmp1 >> tmp2;

        vt[tmp1-1].push\_back(tmp2-1); // dinh tmp2 ke voi dinh tmp1

        vt[tmp2-1].push\_back(tmp1-1); // dinh tmp1 ke voi dinh tmp2

    }

    for(int i=0; i<n; i++){

        visited[i] = false;

    }

    res = 0;

}

bool check(int a, int i){

    if(a == 0) return true;

    if(visited[i]) return false;

    int index = 0;

    for(int j=0; j<vt[x[a-1]].size(); j++){

        if(i == vt[x[a-1]][j]) index++;

    }

    if(index == 0) return false;

    return true;

}

void solution(){

    res++;

}

void TRY(int a){

    for(int i=0; i<n; i++){

        if(check(a, i)){

            visited[i] = true;

            x[a] = i;

            if(a == k) solution();

            else TRY(a+1);

            visited[i] = false;

        }

    }

}

int main(){

    input();

    TRY(0);

    cout << res / 2;

}