Contents

[Bài thực hành số 3 2](#_Toc151760290)

[Bài 3.1. Dãy Lucas được định nghĩa bởi Ln = Ln-1 + Ln-2 với L0 = 2, L1 = 1. Hãy viết hàm tính số Lucas thứ n. 2](#_Toc151760291)

[Bài 3.2. 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). 3](#_Toc151760292)

[Bài 3.3. 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 cij. Hãy tìm tổng chi phí nhỏ nhất có thể 7](#_Toc151760293)

[Bài 3.4. Cho dãy a có n phần tử. Một dãy con của a là dãy thu được bằng cách xóa đi một số phần tử của a 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 a. 11](#_Toc151760294)

[Bài 3.5. Tính hệ số tổ hợp C(n, k) 15](#_Toc151760295)

[Bài 3.6. Tìm ước chung lớn nhất của hai số nguyên a, b cho trước. 19](#_Toc151760296)

[Bài 3.7. Sử dụng phương pháp khử đệ quy bằng stack, hãy liệt kê các xâu nhị phân độ dài n không có k bit 1 nào liên tiếp 21](#_Toc151760297)

[Bài tập 8: Cân đĩa Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng M như người ta nói hay không. Có một cân thăng bằng và n quả cân. Quả thứ i nặng mi . Hãy chỉ ra một cách cân thỏa mãn. Quy cách in ra đã được tích hợp trong mã nguồn dưới 27](#_Toc151760298)

[Bài tập 9: Lập lịch cho y tá. Một y tá cần lập lịch làm việc trong N 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]. Hãy liệt kê tất cả các cách lập lịch tốt, với mỗi lịch in ra trên một dòng một xâu nhị phân độ dài N với bit 0/1 tương ứng là nghỉ/làm việc. Các xâu phải được in ra theo thứ tự từ điển 31](#_Toc151760299)

[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 S là xâu gồm n ký tự 0. Hãy liệt kê tất cả các xâu nhị phân độ dài n, có khoảng cách Hamming với S bằng H. Các xâu phải được liệt kê theo thứ tự từ điển 37](#_Toc151760300)

[Bài tập 11: Lịch trình chụp ảnh Superior là một hòn đảo tuyệt đẹp với n địa điểm chụp ảnh và các đường một chiều nối các điểm chụp ảnh với nhau. Đoàn khách tham quan có r 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. 42](#_Toc151760301)

[Bài tập 12: Đếm đường đi Cho đồ thị vô hướng G, hãy đếm số đường đi đi qua k cạnh và không đi qua đỉnh nào quá một lần. 48](#_Toc151760302)

[Hình 1. Code bài 3.1 hình 1 4](#_Toc151760324)

[Hình 2. Testcase bài 3.1 4](#_Toc151760325)

[Hình 3. Code bài 3.2 hình 1 5](#_Toc151760326)

[Hình 4. Code bài 3.2 hình 2 6](#_Toc151760327)

[Hình 5. Code bài 3.2 hình 3 6](#_Toc151760328)

[Hình 6. Testcase bài 3.2 hình 1 7](#_Toc151760329)

[Hình 7. Testcase bài 3.2 hình 2 7](#_Toc151760330)

[Hình 8. Code bài 3.3 hình 1 10](#_Toc151760331)

[Hình 9. Code bài 3.3 hình 2 10](#_Toc151760332)

[Hình 10. Code bài 3.3 hình 3 10](#_Toc151760333)

[Hình 11. Testcase bài 3.3 11](#_Toc151760334)

[Hình 12. Đề bài 3.4 13](#_Toc151760335)

[Hình 13. Code bài 3.4 hình 1 14](#_Toc151760336)

[Hình 14. Code bài 3.4 hình 2 14](#_Toc151760337)

[Hình 15. Code bài 3.4 hình 3 14](#_Toc151760338)

[Hình 16. Testcase bài 3.4 15](#_Toc151760339)

[Hình 17. Đề bài 3.5 17](#_Toc151760340)

[Hình 18. Code bài 3.5 hình 1 18](#_Toc151760341)

[Hình 19. Code bài 3.5 hình 2 18](#_Toc151760342)

[Hình 20. Testcase bài 3.5 hình 1 18](#_Toc151760343)

[Hình 21. Testcase bài 3.5 hình 2 19](#_Toc151760344)

[Hình 22. Testcase bài 3.5 hình 3 19](#_Toc151760345)

[Hình 23. Testcase bài 3.5 hình 4 20](#_Toc151760346)

[Hình 24. Đề bài 3.6 22](#_Toc151760347)

[Hình 25. Code bài 3.6 hình 1 22](#_Toc151760348)

[Hình 26. Code bài 3.6 hình 2 22](#_Toc151760349)

[Hình 27. Testcase bài 3.6 23](#_Toc151760350)

[Hình 28. Đề bài 3.7 24](#_Toc151760351)

[Hình 29. Code bài 3.7 hình 1 25](#_Toc151760352)

[Hình 30. Code bài 3.7 hình 2 25](#_Toc151760353)

[Hình 31. Code bài 3.7 hình 3 25](#_Toc151760354)

[Hình 32. Testcase bài 3.7 hình 1 26](#_Toc151760355)

[Hình 33. Testcase bài 3.7 hình 2 26](#_Toc151760356)

[Hình 34. Testcase bài 3.7 hình 3 27](#_Toc151760357)

[Hình 35. Testcase bài 3.7 hình 4 27](#_Toc151760358)

[Hình 36. Code bài 3.8 hình 1 30](#_Toc151760359)

[Hình 37. Code bài 3.8 hình 2 30](#_Toc151760360)

[Hình 38. Code bài 3.8 hình 3 31](#_Toc151760361)

[Hình 39. Output bài 3.8 31](#_Toc151760362)

[Hình 40. Code bài 3.9 hình 1 34](#_Toc151760363)

[Hình 41. Code bài 3.9 hình 2 34](#_Toc151760364)

[Hình 42. Code bài 3.9 hình 3 35](#_Toc151760365)

[Hình 43. Code bài 3.9 hình 4 35](#_Toc151760366)

[Hình 44. Output testcase 1 bài 3.9 35](#_Toc151760367)

[Hình 45. Output testcase 2 bài 3.9 36](#_Toc151760368)

[Hình 46. Output testcase 3 bài 3.9 36](#_Toc151760369)

[Hình 47. Output testcase 4 bài 3.9 37](#_Toc151760370)

[Hình 48. Output testcase 5 bài 3.9 37](#_Toc151760371)

[Hình 49. Code bài 3.10 hình 1 41](#_Toc151760372)

[Hình 50. Code bài 3.10 hình 2 41](#_Toc151760373)

[Hình 51. Code bài 3.10 hình 3 41](#_Toc151760374)

[Hình 52. Output testcase 1 bài 3.10 hình 1 42](#_Toc151760375)

[Hình 53. Output testcase 1 bài 3.10 hình 2 42](#_Toc151760376)

[Hình 54. Output testcase 1 bài 3.10 hình 3 43](#_Toc151760377)

[Hình 55. Code bài 3.11 hình 1 45](#_Toc151760378)

[Hình 56. Code bài 3.11 hình 2 46](#_Toc151760379)

[Hình 57. Code bài 3.11 hình 3 46](#_Toc151760380)

[Hình 58. Code bài 3.11 hình 4 47](#_Toc151760381)

[Hình 59. Output testcase 1 bài 3.11 47](#_Toc151760382)

[Hình 60. Output testcase 2 bài 3.11 47](#_Toc151760383)

[Hình 61. Code bài 3.12 hình 1 51](#_Toc151760384)

[Hình 62. Code bài 3.12 hình 2 51](#_Toc151760385)

[Hình 63. Code bài 3.12 hình 3 52](#_Toc151760386)

[Hình 64. Output testcase 1 bài 3.12 52](#_Toc151760387)

[Hình 65. Output testcase 2 bài 3.12 52](#_Toc151760388)

[Hình 66. Output testcase 3 bài 3.12 53](#_Toc151760389)

[Hình 67. Output testcase 4 bài 3.12 53](#_Toc151760390)

[Hình 68. Output testcase 5 bài 3.12 53](#_Toc151760391)

[Hình 69. Output testcase 6 bài 3.12 53](#_Toc151760392)

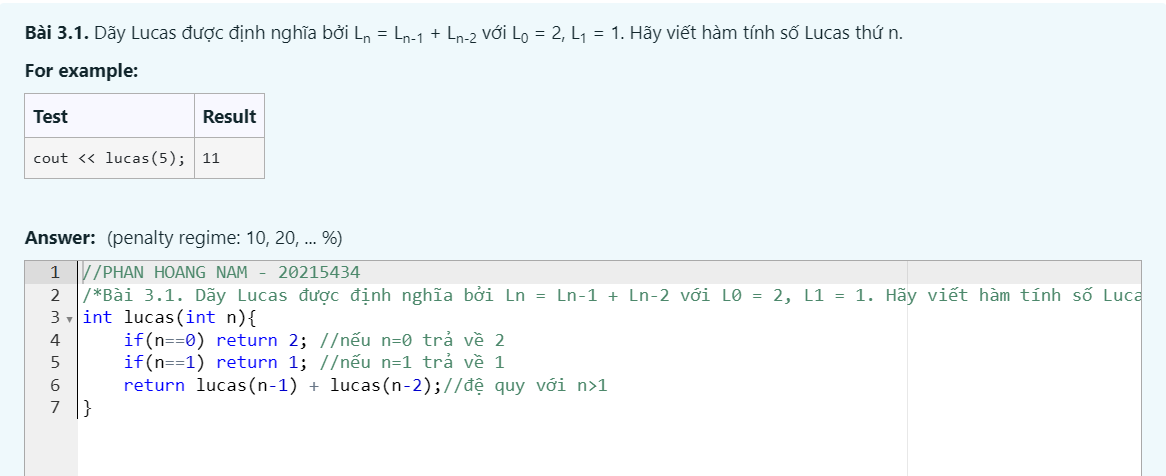
[Hình 70. Output testcase 7 bài 3.12 54](#_Toc151760393)

[Hình 71. Output testcase 8 bài 3.12 54](#_Toc151760394)

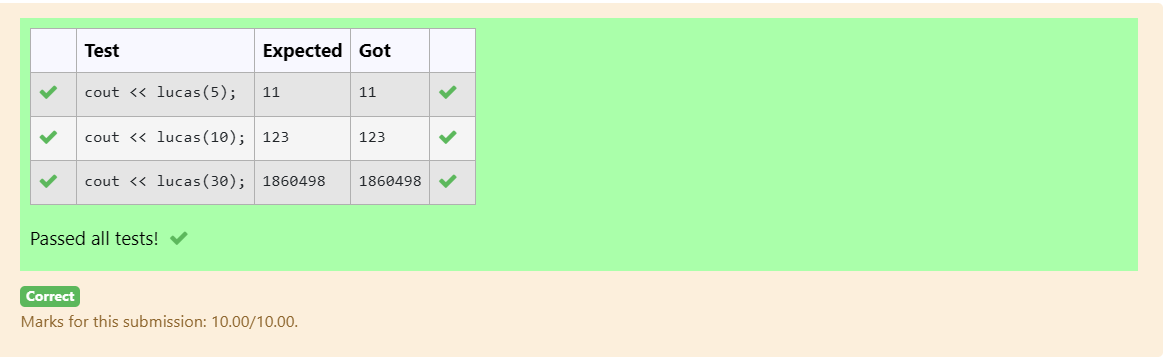
[Hình 72. Output testcase 9 bài 3.12 54](#_Toc151760395)

# Bài thực hành số 3

## Bài 3.1. Dãy Lucas được định nghĩa bởi Ln = Ln-1 + Ln-2 với L0 = 2, L1 = 1. Hãy viết hàm tính số Lucas thứ n.



Hình . Code bài 3.1 hình 1



Hình . Testcase bài 3.1

//PHAN HOANG NAM - 20215434

/\*Bài 3.1. Dãy Lucas được định nghĩa bởi Ln = Ln-1 + Ln-2 với L0 = 2, L1 = 1. Hãy viết hàm tính số Lucas thứ n.\*/

int lucas(int n){

if(n==0) return 2; //nếu n=0 trả về 2

if(n==1) return 1; //nếu n=1 trả về 1

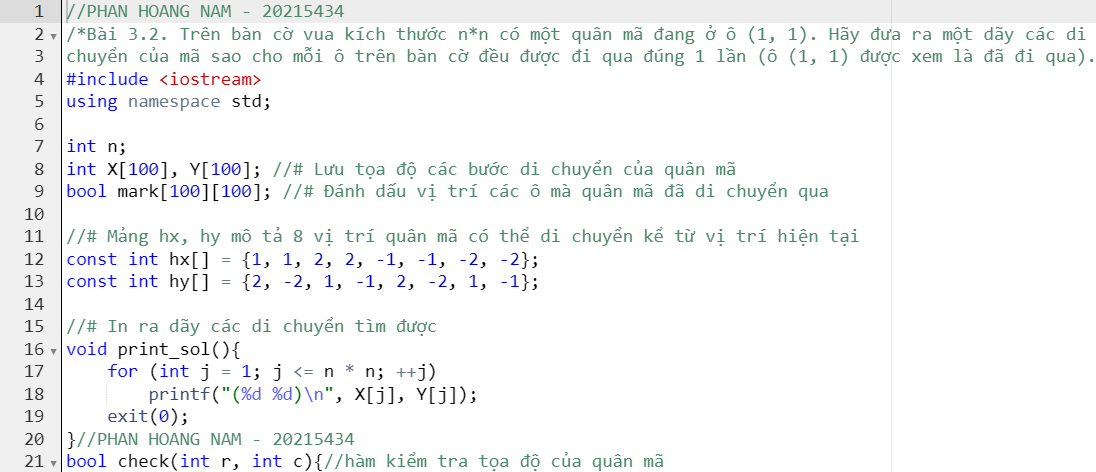
return lucas(n-1) + lucas(n-2);//đệ quy với n>1

}

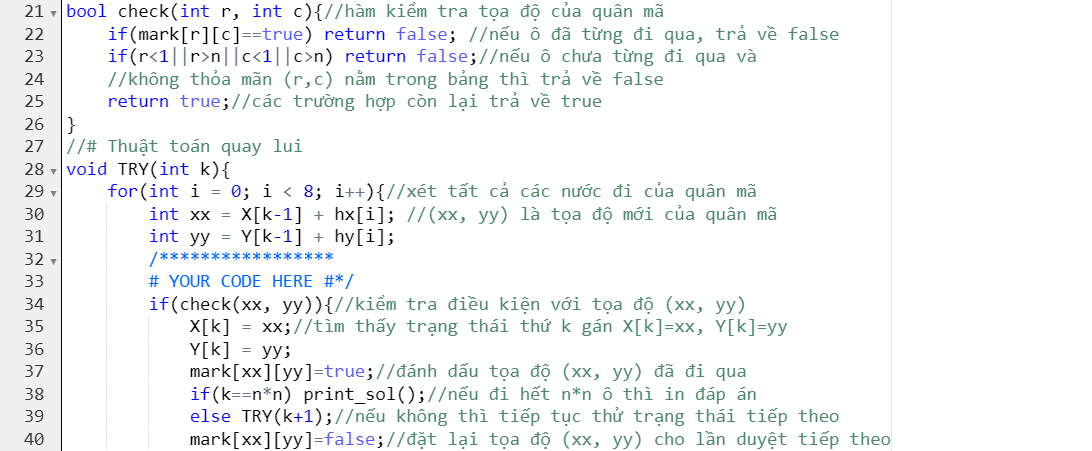
## Bài 3.2. Trên bàn cờ vua kích thước n\*n có một quân mã đang ở ô (1, 1). Hãy đưa ra một dãy các di chuyển của mã sao cho mỗi ô trên bàn cờ đều được đi qua đúng 1 lần (ô (1, 1) được xem là đã đi qua).



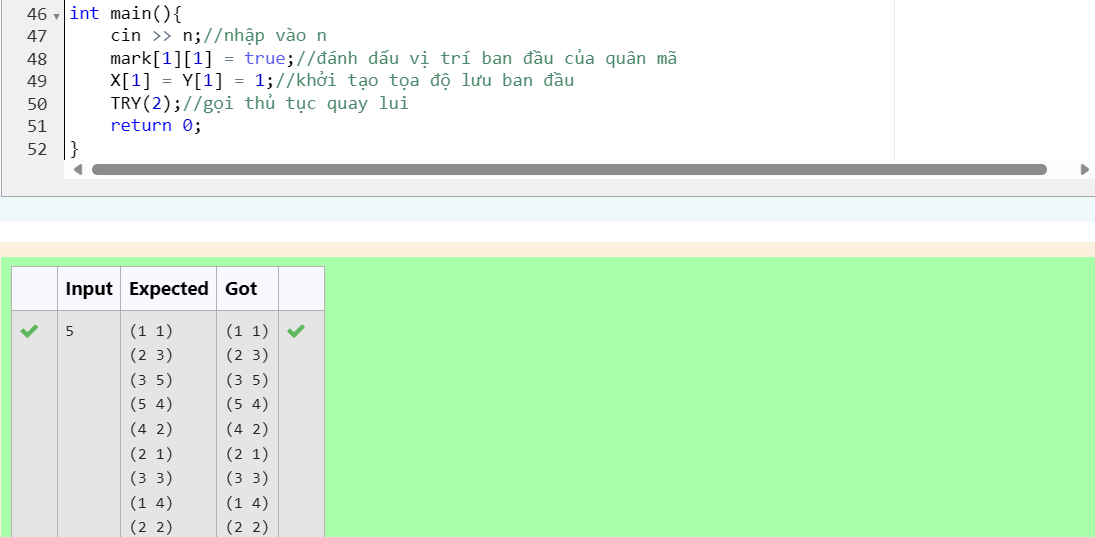
Hình . Code bài 3.2 hình 1



Hình . Code bài 3.2 hình 2



Hình . Code bài 3.2 hình 3



Hình . Testcase bài 3.2 hình 1



Hình . Testcase bài 3.2 hình 2

//PHAN HOANG NAM - 20215434

/\*Bài 3.2. 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).\*/

#include <iostream>

using namespace std;

int n;

int X[100], Y[100]; //# Lưu tọa độ các bước di chuyển của quân mã

bool 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);

}//PHAN HOANG NAM - 20215434

bool check(int r, int c){//hàm kiểm tra tọa độ của quân mã

if(mark[r][c]==true) return false; //nếu ô đã từng đi qua, trả về false

if(r<1||r>n||c<1||c>n) return false;//nếu ô chưa từng đi qua và

//không thỏa mãn (r,c) nằm trong bảng thì trả về false

return true;//các trường hợp còn lại trả về true

}

//# Thuật toán quay lui

void TRY(int k){

for(int i = 0; i < 8; i++){//xét tất cả các nước đi của quân mã

int xx = X[k-1] + hx[i]; //(xx, yy) là tọa độ mới của quân mã

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

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

# YOUR CODE HERE #\*/

if(check(xx, yy)){//kiểm tra điều kiện với tọa độ (xx, yy)

X[k] = xx;//tìm thấy trạng thái thứ k gán X[k]=xx, Y[k]=yy

Y[k] = yy;

mark[xx][yy]=true;//đánh dấu tọa độ (xx, yy) đã đi qua

if(k==n\*n) print\_sol();//nếu đi hết n\*n ô thì in đáp án

else TRY(k+1);//nếu không thì tiếp tục thử trạng thái tiếp theo

mark[xx][yy]=false;//đặt lại tọa độ (xx, yy) cho lần duyệt tiếp theo

}//PHAN HOANG NAM - 20215434

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

}

}

int main(){

cin >> n;//nhập vào n

mark[1][1] = true;//đánh dấu vị trí ban đầu của quân mã

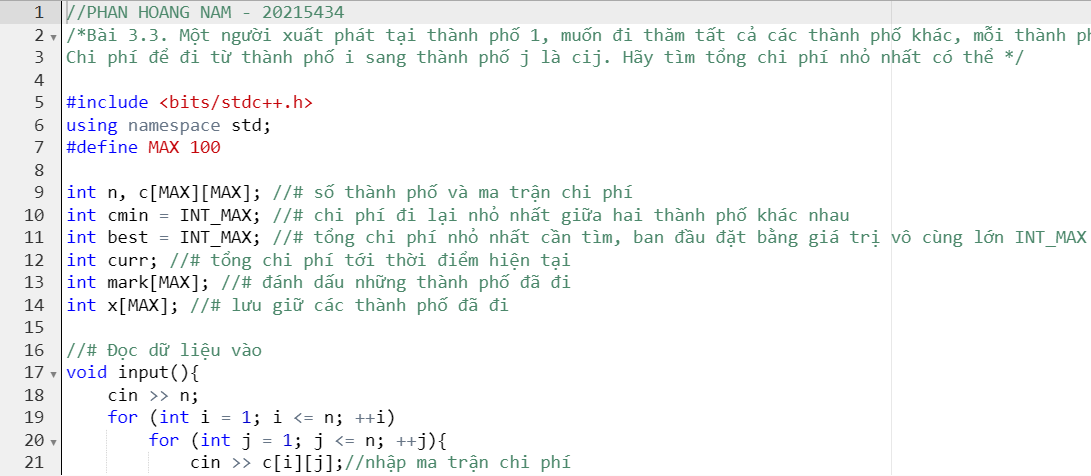
X[1] = Y[1] = 1;//khởi tạo tọa độ lưu ban đầu

TRY(2);//gọi thủ tục quay lui

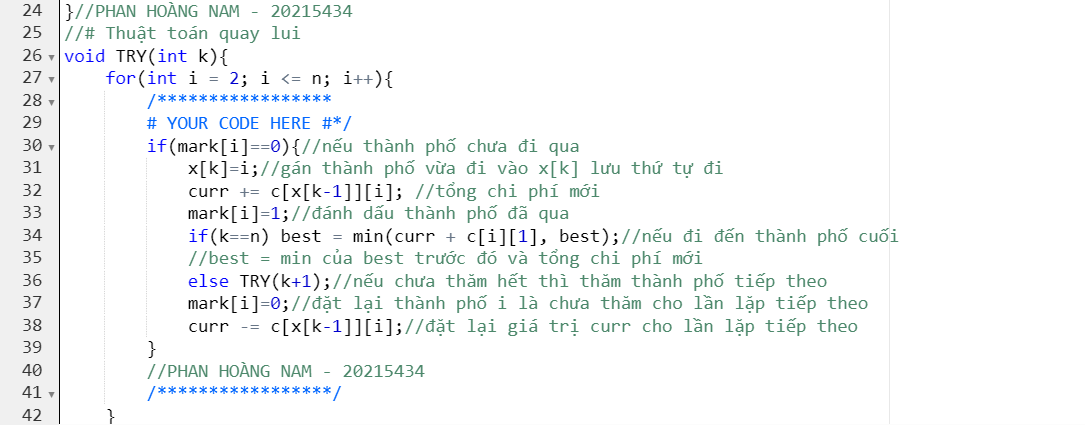
return 0;

}

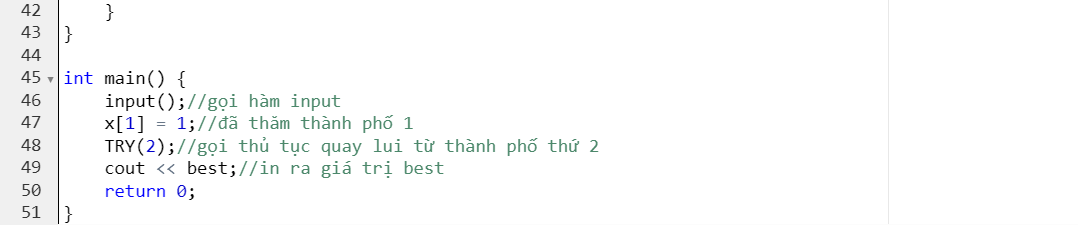
## Bài 3.3. Một người xuất phát tại thành phố 1, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng 1 lần và quay về 1. Chi phí để đi từ thành phố i sang thành phố j là cij. Hãy tìm tổng chi phí nhỏ nhất có thể



Hình . Code bài 3.3 hình 1



Hình . Code bài 3.3 hình 2



Hình . Code bài 3.3 hình 3



Hình . Testcase bài 3.3

//PHAN HOANG NAM - 20215434

/\*Bài 3.3. 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 cij. Hãy tìm tổng chi phí nhỏ nhất có thể \*/

#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];//nhập ma trận chi phí

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

}

}//PHAN HOÀNG NAM - 20215434

//# Thuật toán quay lui

void TRY(int k){

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

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

# YOUR CODE HERE #\*/

if(mark[i]==0){//nếu thành phố chưa đi qua

x[k]=i;//gán thành phố vừa đi vào x[k] lưu thứ tự đi

curr += c[x[k-1]][i]; //tổng chi phí mới

mark[i]=1;//đánh dấu thành phố đã qua

if(k==n) best = min(curr + c[i][1], best);//nếu đi đến thành phố cuối

//best = min của best trước đó và tổng chi phí mới

else TRY(k+1);//nếu chưa thăm hết thì thăm thành phố tiếp theo

mark[i]=0;//đặt lại thành phố i là chưa thăm cho lần lặp tiếp theo

curr -= c[x[k-1]][i];//đặt lại giá trị curr cho lần lặp tiếp theo

}

//PHAN HOÀNG NAM - 20215434

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

}

}

int main() {

input();//gọi hàm input

x[1] = 1;//đã thăm thành phố 1

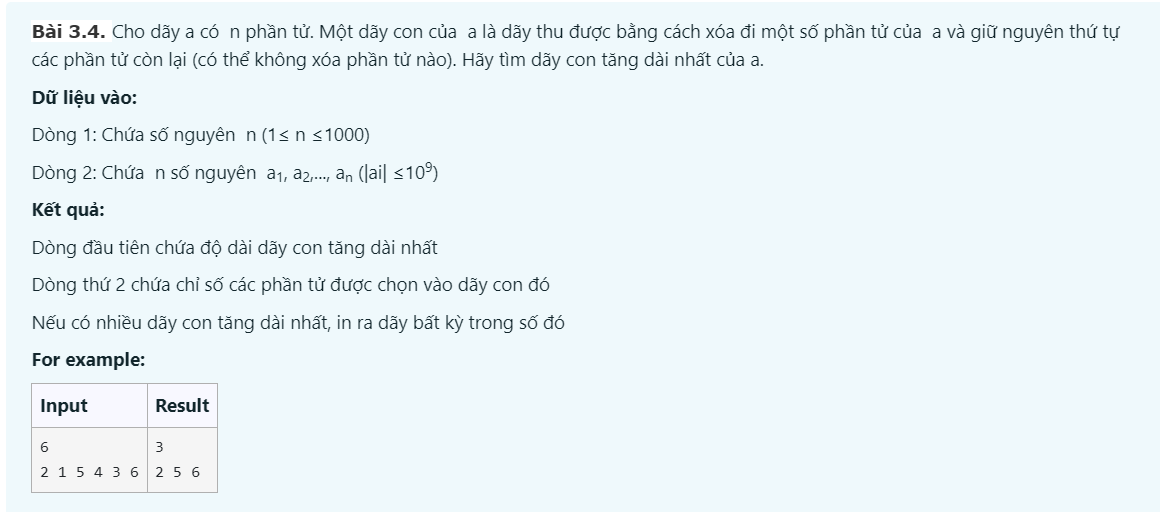
TRY(2);//gọi thủ tục quay lui từ thành phố thứ 2

cout << best;//in ra giá trị best

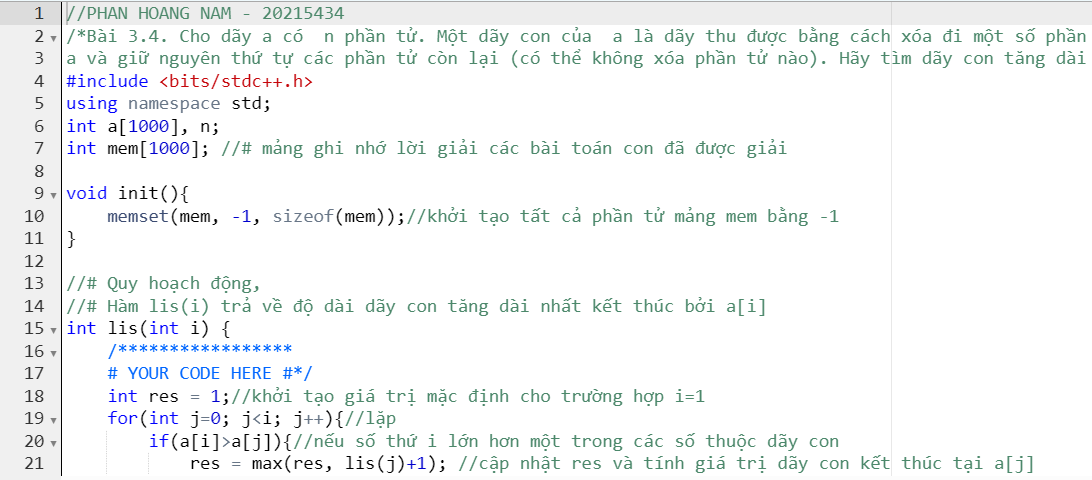
return 0;

}

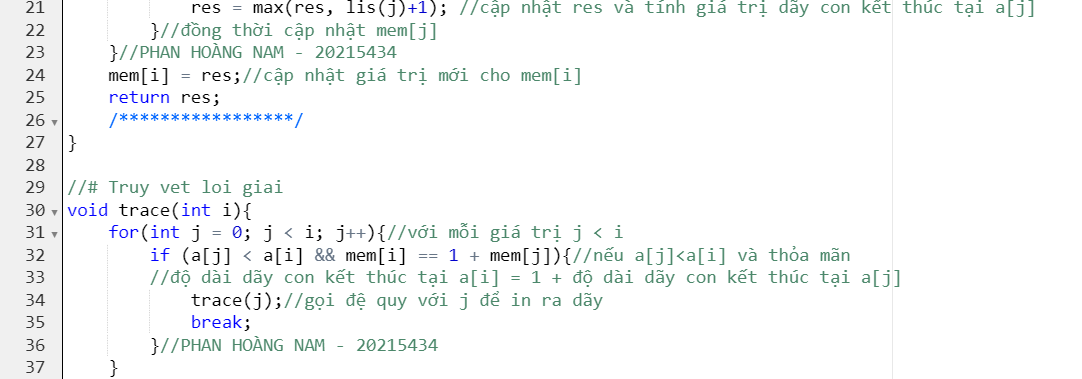
## Bài 3.4. Cho dãy a có n phần tử. Một dãy con của a là dãy thu được bằng cách xóa đi một số phần tử của a và giữ nguyên thứ tự các phần tử còn lại (có thể không xóa phần tử nào). Hãy tìm dãy con tăng dài nhất của a.



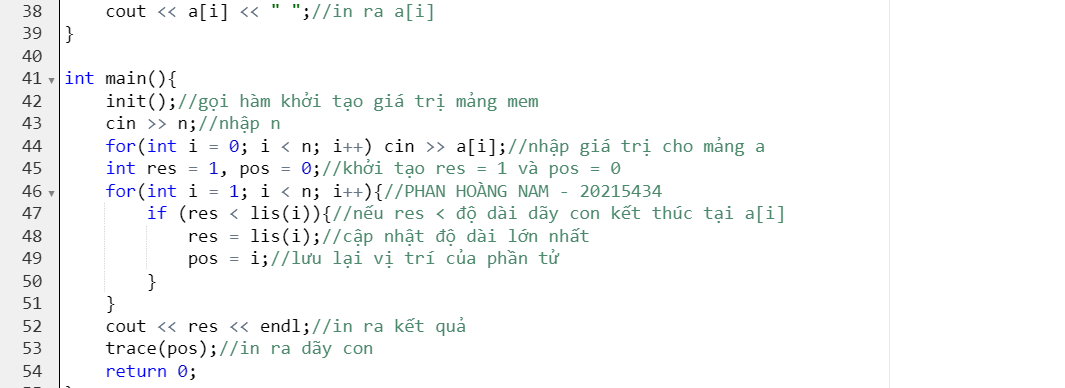
Hình . Đề bài 3.4



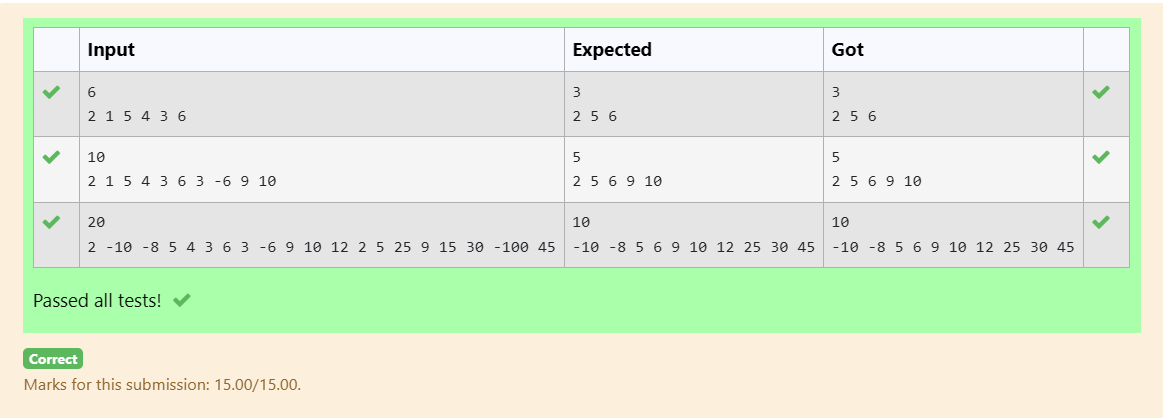
Hình . Code bài 3.4 hình 1



Hình . Code bài 3.4 hình 2



Hình . Code bài 3.4 hình 3



Hình . Testcase bài 3.4

//PHAN HOANG NAM - 20215434

/\*Bài 3.4. Cho dãy a có n phần tử. Một dãy con của a là dãy thu được bằng cách xóa đi một số phần tử của

a 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 a.\*/

#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, -1, sizeof(mem));//khởi tạo tất cả phần tử mảng mem bằng -1

}

//# Quy hoạch động,

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

int lis(int i) {

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

# YOUR CODE HERE #\*/

int res = 1;//khởi tạo giá trị mặc định cho trường hợp i=1

for(int j=0; j<i; j++){//lặp

if(a[i]>a[j]){//nếu số thứ i lớn hơn một trong các số thuộc dãy con

res = max(res, lis(j)+1); //cập nhật res và tính giá trị dãy con kết thúc tại a[j]

}//đồng thời cập nhật mem[j]

}//PHAN HOÀNG NAM - 20215434

mem[i] = res;//cập nhật giá trị mới cho mem[i]

return res;

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

}

//# Truy vet loi giai

void trace(int i){

for(int j = 0; j < i; j++){//với mỗi giá trị j < i

if (a[j] < a[i] && mem[i] == 1 + mem[j]){//nếu a[j]<a[i] và thỏa mãn

//độ dài dãy con kết thúc tại a[i] = 1 + độ dài dãy con kết thúc tại a[j]

trace(j);//gọi đệ quy với j để in ra dãy

break;

}//PHAN HOÀNG NAM - 20215434

}

cout << a[i] << " ";//in ra a[i]

}

int main(){

init();//gọi hàm khởi tạo giá trị mảng mem

cin >> n;//nhập n

for(int i = 0; i < n; i++) cin >> a[i];//nhập giá trị cho mảng a

int res = 1, pos = 0;//khởi tạo res = 1 và pos = 0

for(int i = 1; i < n; i++){//PHAN HOÀNG NAM - 20215434

if (res < lis(i)){//nếu res < độ dài dãy con kết thúc tại a[i]

res = lis(i);//cập nhật độ dài lớn nhất

pos = i;//lưu lại vị trí của phần tử

}

}

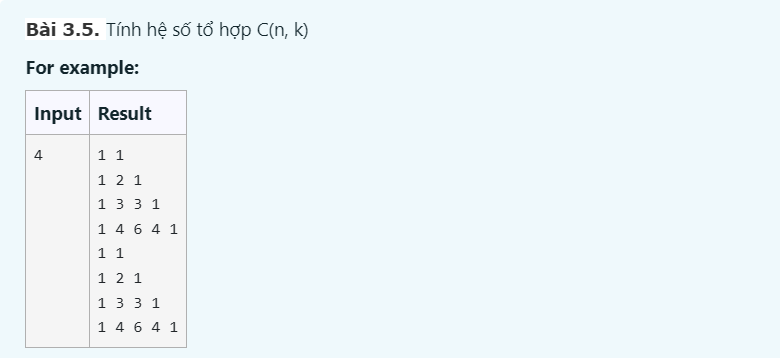
cout << res << endl;//in ra kết quả

trace(pos);//in ra dãy con

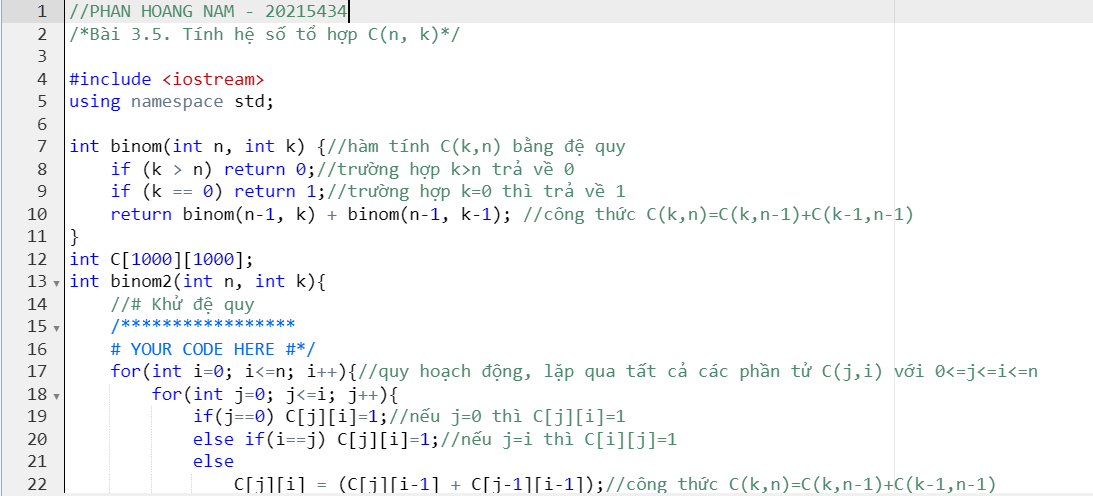
return 0;

}

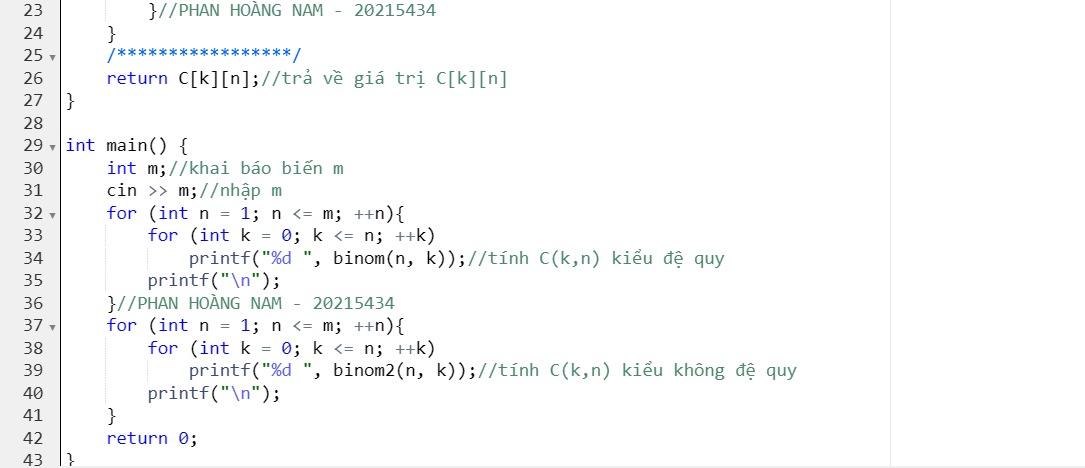
## Bài 3.5. Tính hệ số tổ hợp C(n, k)



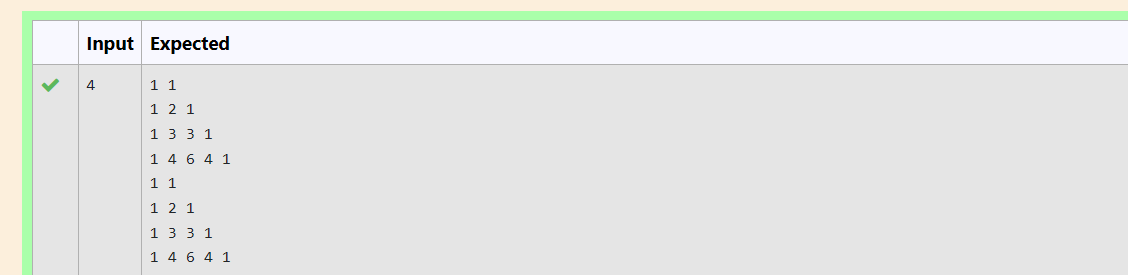
Hình . Đề bài 3.5



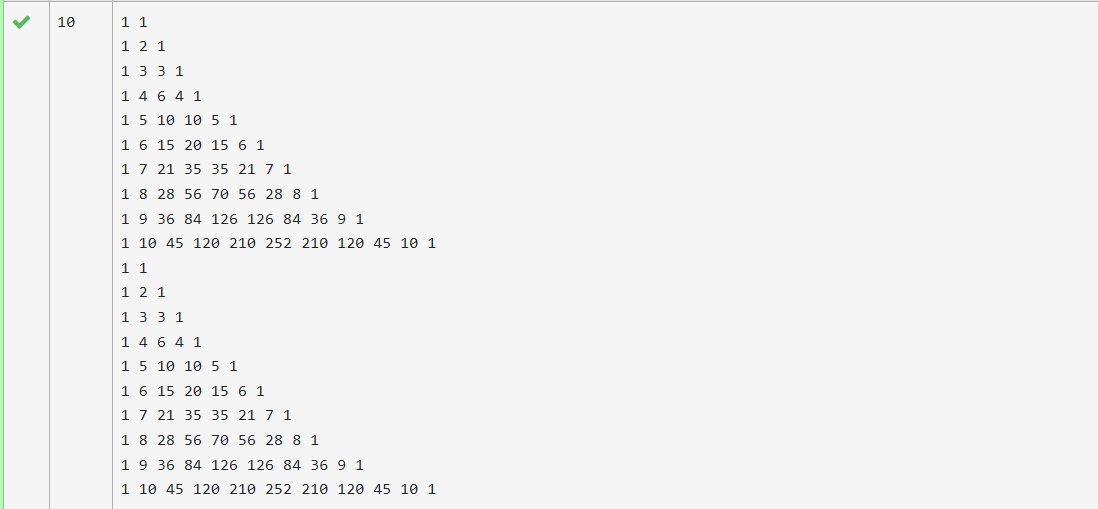
Hình . Code bài 3.5 hình 1



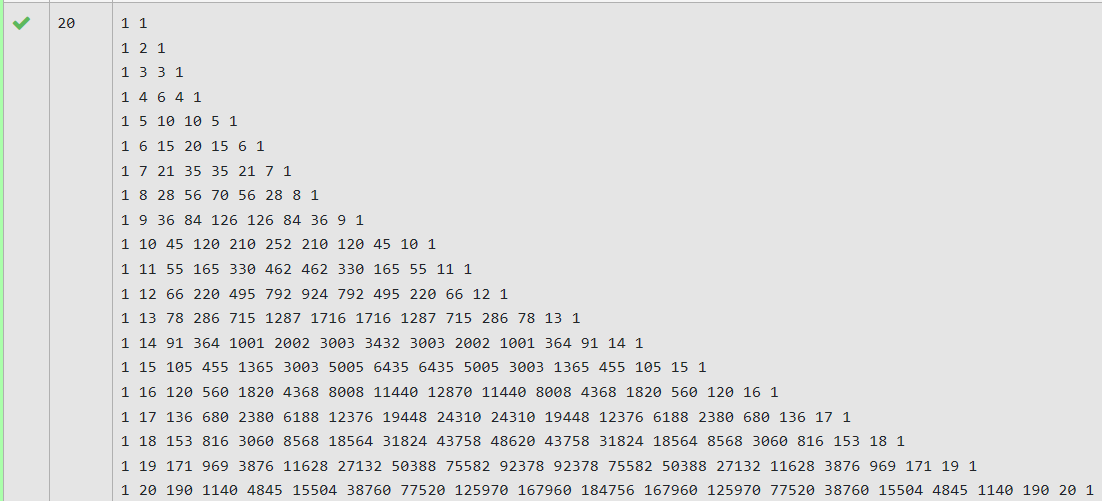
Hình . Code bài 3.5 hình 2



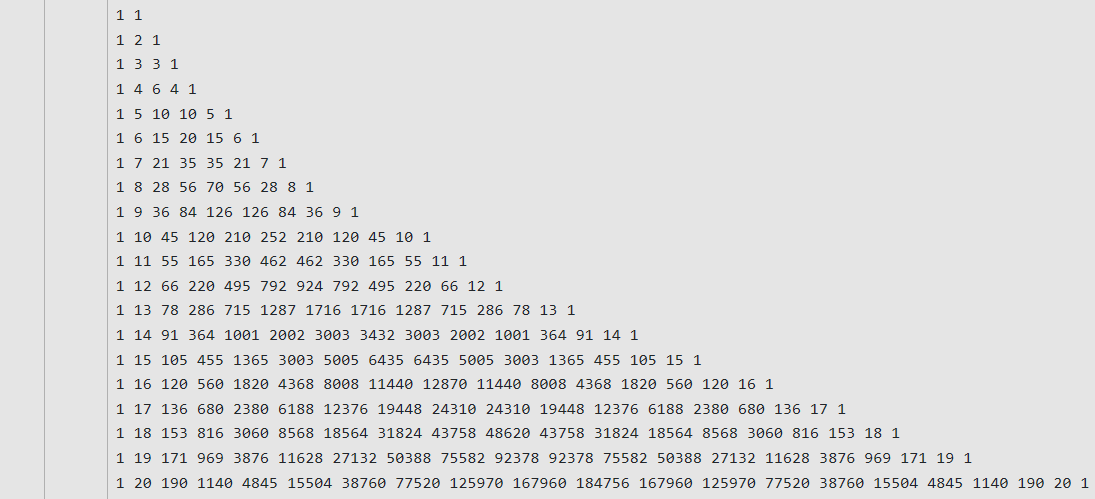
Hình . Testcase bài 3.5 hình 1



Hình . Testcase bài 3.5 hình 2



Hình . Testcase bài 3.5 hình 3



Hình . Testcase bài 3.5 hình 4

//PHAN HOANG NAM - 20215434

/\*Bài 3.5. Tính hệ số tổ hợp C(n, k)\*/

#include <iostream>

using namespace std;

int binom(int n, int k) {//hàm tính C(k,n) bằng đệ quy

if (k > n) return 0;//trường hợp k>n trả về 0

if (k == 0) return 1;//trường hợp k=0 thì trả về 1

return binom(n-1, k) + binom(n-1, k-1); //công thức C(k,n)=C(k,n-1)+C(k-1,n-1)

}

int C[1000][1000];

int binom2(int n, int k){

//# Khử đệ quy

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

# YOUR CODE HERE #\*/

for(int i=0; i<=n; i++){//quy hoạch động, lặp qua tất cả các phần tử C(j,i) với 0<=j<=i<=n

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

if(j==0) C[j][i]=1;//nếu j=0 thì C[j][i]=1

else if(i==j) C[j][i]=1;//nếu j=i thì C[i][j]=1

else

C[j][i] = (C[j][i-1] + C[j-1][i-1]);//công thức C(k,n)=C(k,n-1)+C(k-1,n-1)

}//PHAN HOÀNG NAM - 20215434

}

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

return C[k][n];//trả về giá trị C[k][n]

}

int main() {

int m;//khai báo biến m

cin >> m;//nhập m

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

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

printf("%d ", binom(n, k));//tính C(k,n) kiểu đệ quy

printf("\n");

}//PHAN HOÀNG NAM - 20215434

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

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

printf("%d ", binom2(n, k));//tính C(k,n) kiểu không đệ quy

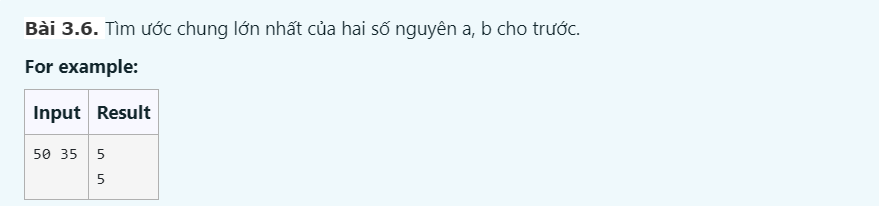
printf("\n");

}

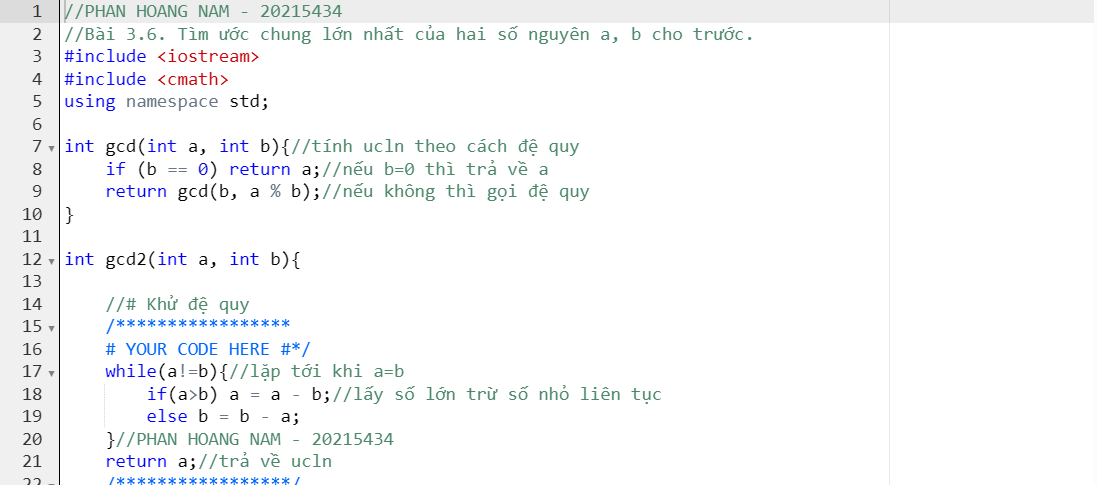
return 0;

}

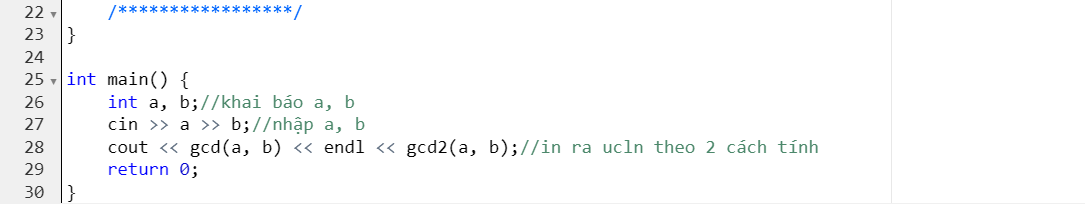
## Bài 3.6. Tìm ước chung lớn nhất của hai số nguyên a, b cho trước.



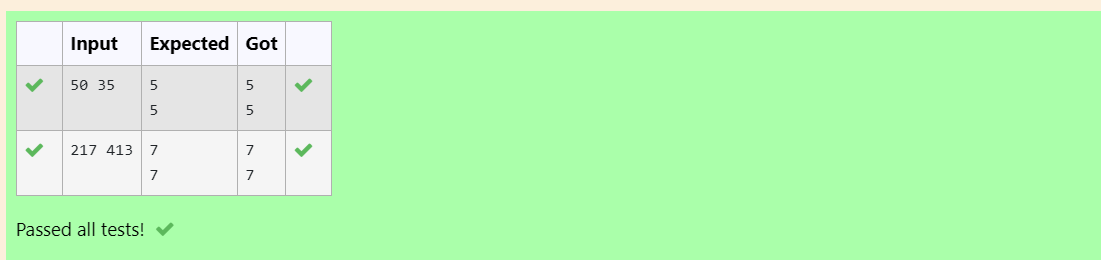
Hình . Đề bài 3.6



Hình . Code bài 3.6 hình 1



Hình . Code bài 3.6 hình 2



Hình . Testcase bài 3.6

//PHAN HOANG NAM - 20215434

//Bài 3.6. Tìm ước chung lớn nhất của hai số nguyên a, b cho trước.

#include <iostream>

#include <cmath>

using namespace std;

int gcd(int a, int b){//tính ucln theo cách đệ quy

if (b == 0) return a;//nếu b=0 thì trả về a

return gcd(b, a % b);//nếu không thì gọi đệ quy

}

int gcd2(int a, int b){

//# Khử đệ quy

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

# YOUR CODE HERE #\*/

while(a!=b){//lặp tới khi a=b

if(a>b) a = a - b;//lấy số lớn trừ số nhỏ liên tục

else b = b - a;

}//PHAN HOANG NAM - 20215434

return a;//trả về ucln

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

}

int main() {

int a, b;//khai báo a, b

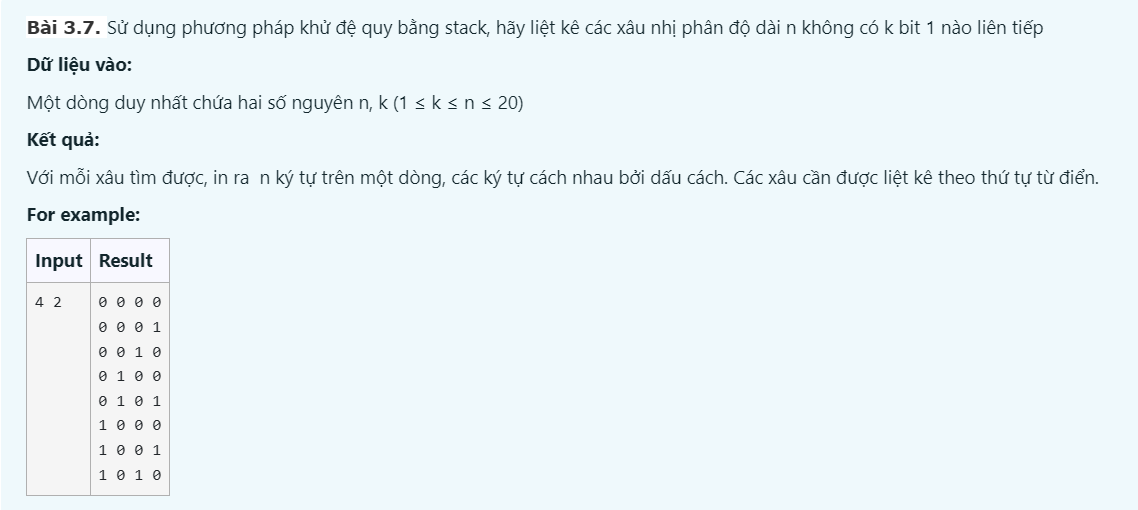
cin >> a >> b;//nhập a, b

cout << gcd(a, b) << endl << gcd2(a, b);//in ra ucln theo 2 cách tính

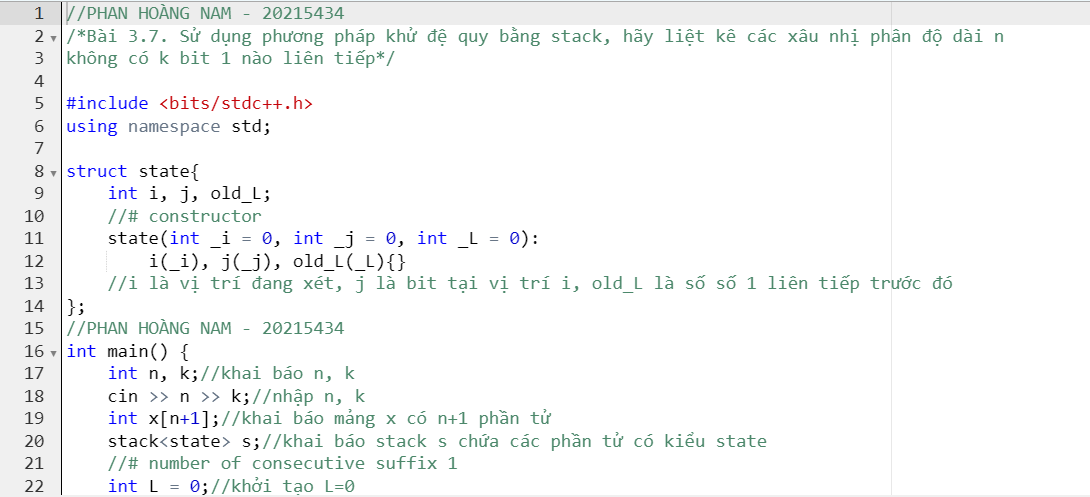
return 0;

}

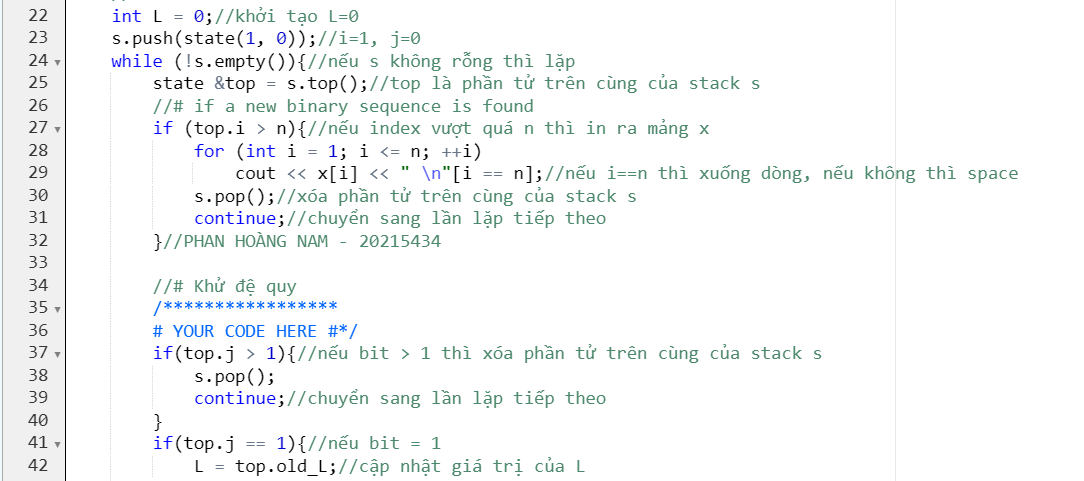
## Bài 3.7. Sử dụng phương pháp khử đệ quy bằng stack, hãy liệt kê các xâu nhị phân độ dài n không có k bit 1 nào liên tiếp



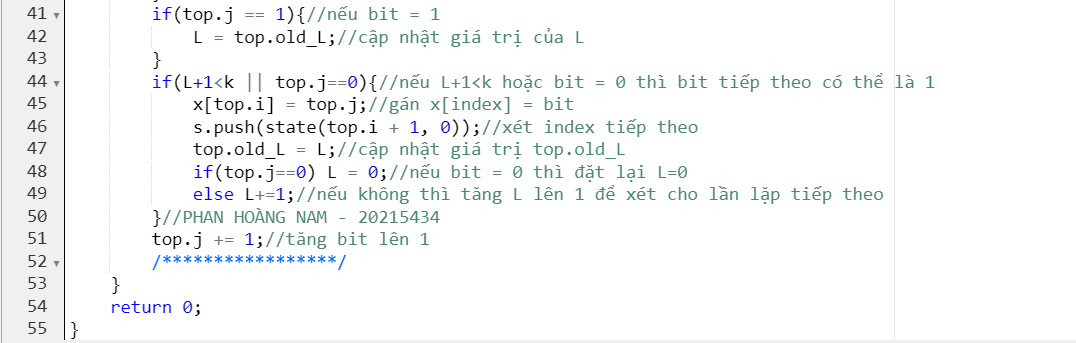
Hình . Đề bài 3.7



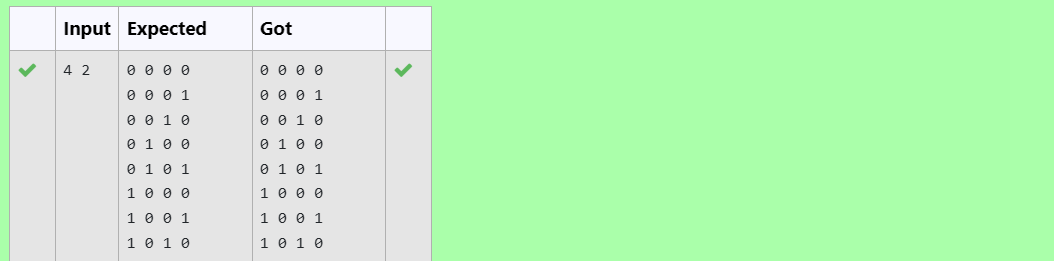
Hình . Code bài 3.7 hình 1



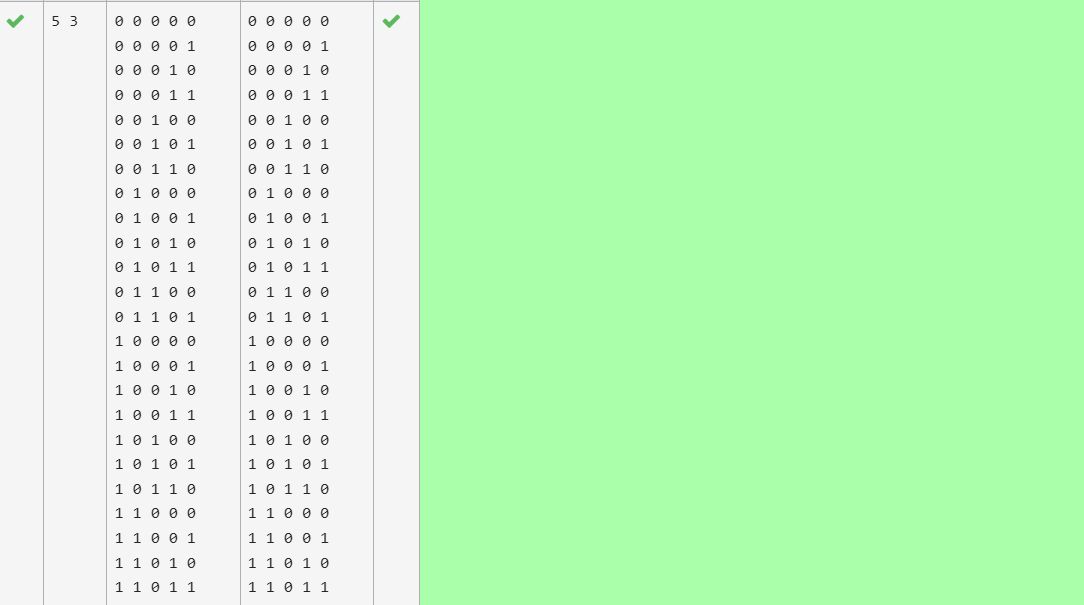
Hình . Code bài 3.7 hình 2



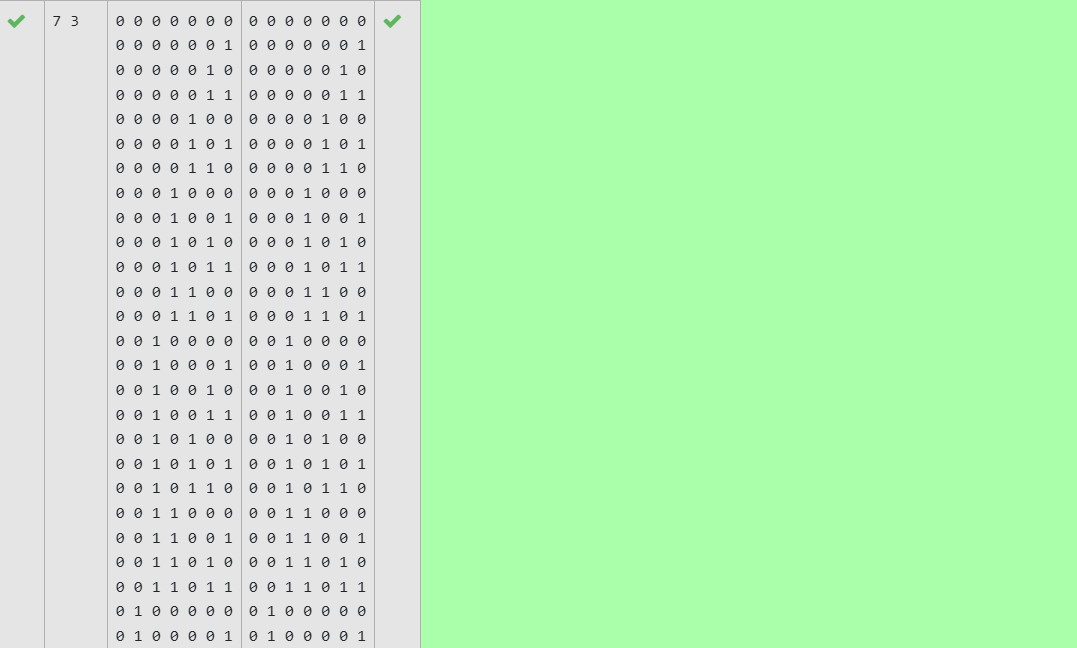
Hình . Code bài 3.7 hình 3



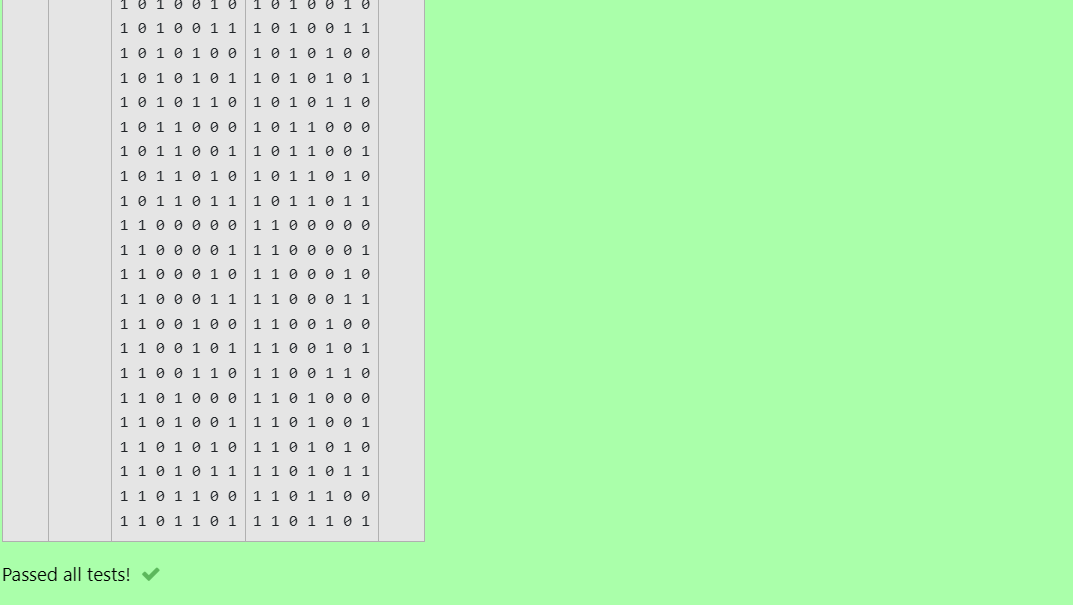
Hình . Testcase bài 3.7 hình 1



Hình . Testcase bài 3.7 hình 2



Hình . Testcase bài 3.7 hình 3



Hình . Testcase bài 3.7 hình 4

//PHAN HOÀNG NAM - 20215434

/\*Bài 3.7. Sử dụng phương pháp khử đệ quy bằng stack, hãy liệt kê các xâu nhị phân độ dài n

không có k bit 1 nào liên tiếp\*/

#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){}

//i là vị trí đang xét, j là bit tại vị trí i, old\_L là số số 1 liên tiếp trước đó

};

//PHAN HOÀNG NAM - 20215434

int main() {

int n, k;//khai báo n, k

cin >> n >> k;//nhập n, k

int x[n+1];//khai báo mảng x có n+1 phần tử

stack<state> s;//khai báo stack s chứa các phần tử có kiểu state

//# number of consecutive suffix 1

int L = 0;//khởi tạo L=0

s.push(state(1, 0));//i=1, j=0

while (!s.empty()){//nếu s không rỗng thì lặp

state &top = s.top();//top là phần tử trên cùng của stack s

//# if a new binary sequence is found

if (top.i > n){//nếu index vượt quá n thì in ra mảng x

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

cout << x[i] << " \n"[i == n];//nếu i==n thì xuống dòng, nếu không thì space

s.pop();//xóa phần tử trên cùng của stack s

continue;//chuyển sang lần lặp tiếp theo

}//PHAN HOÀNG NAM - 20215434

//# Khử đệ quy

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

# YOUR CODE HERE #\*/

if(top.j > 1){//nếu bit > 1 thì xóa phần tử trên cùng của stack s

s.pop();

continue;//chuyển sang lần lặp tiếp theo

}

if(top.j == 1){//nếu bit = 1

L = top.old\_L;//cập nhật giá trị của L

}

if(L+1<k || top.j==0){//nếu L+1<k hoặc bit = 0 thì bit tiếp theo có thể là 1

x[top.i] = top.j;//gán x[index] = bit

s.push(state(top.i + 1, 0));//xét index tiếp theo

top.old\_L = L;//cập nhật giá trị top.old\_L

if(top.j==0) L = 0;//nếu bit = 0 thì đặt lại L=0

else L+=1;//nếu không thì tăng L lên 1 để xét cho lần lặp tiếp theo

}//PHAN HOÀNG NAM - 20215434

top.j += 1;//tăng bit lên 1

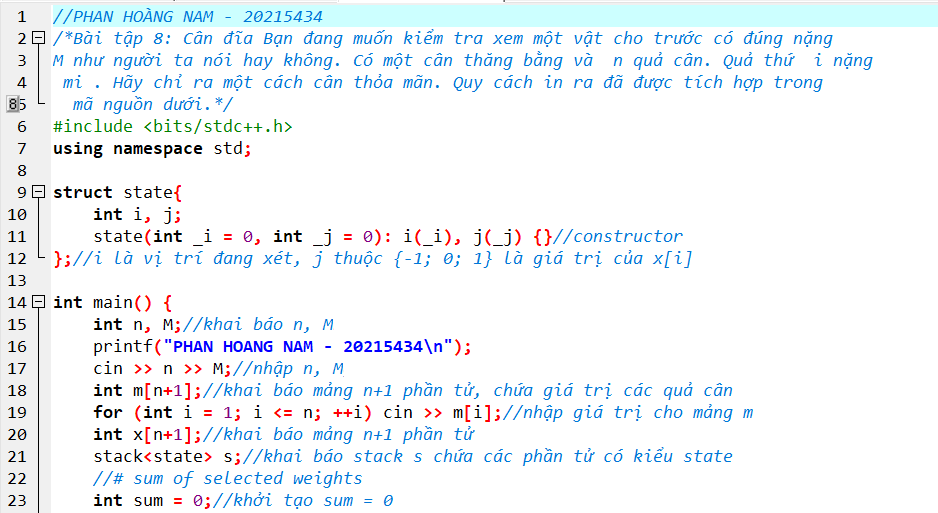
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

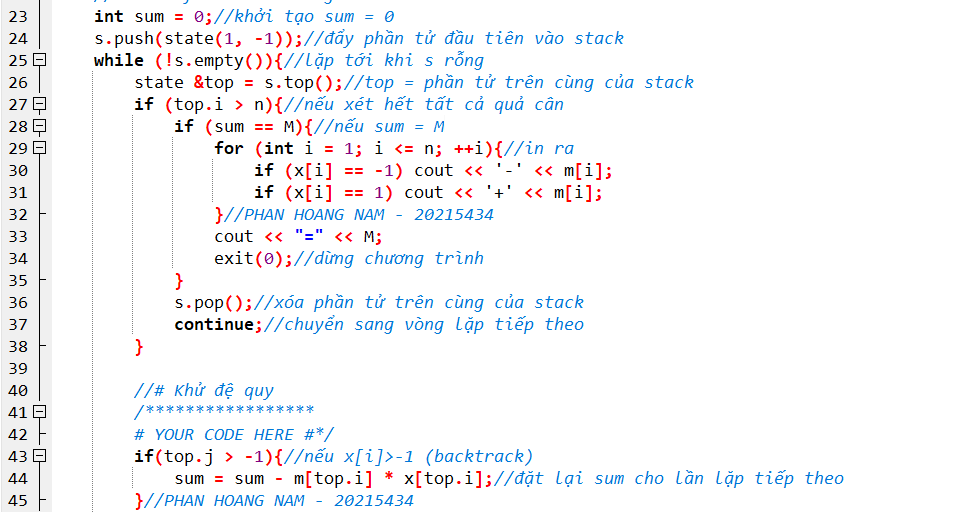
return 0;

}

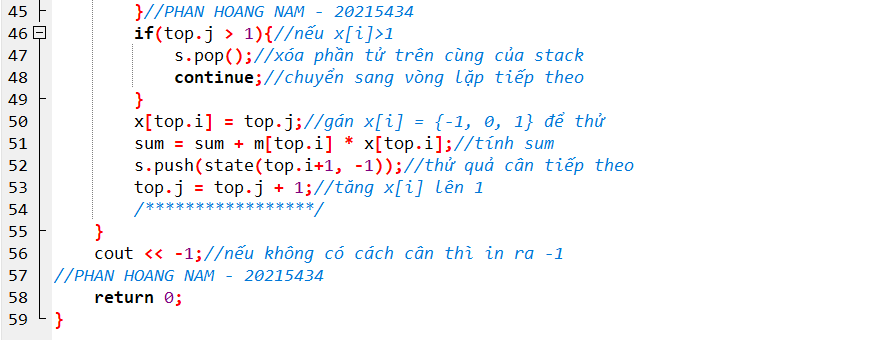
Bài tập 8: Cân đĩa Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng M như người ta nói hay không. Có một cân thăng bằng và n quả cân. Quả thứ i nặng mi . Hãy chỉ ra một cách cân thỏa mãn. Quy cách in ra đã được tích hợp trong mã nguồn dưới.



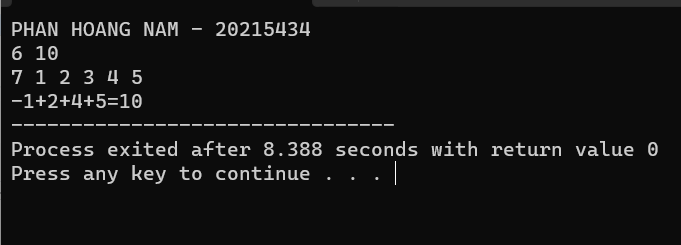
Hình . Code bài 3.8 hình 1



Hình . Code bài 3.8 hình 2



Hình . Code bài 3.8 hình 3



Hình . Output bài 3.8

//PHAN HOÀNG NAM - 20215434

/\*Bài tập 8: Cân đĩa Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng

M như người ta nói hay không. Có một cân thăng bằng và n quả cân. Quả thứ i nặng

mi . Hãy chỉ ra một cách cân thỏa mãn. Quy cách in ra đã được tích hợp trong

mã nguồn dưới.\*/

#include <bits/stdc++.h>

using namespace std;

struct state{

int i, j;

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

};//i là vị trí đang xét, j thuộc {-1; 0; 1} là giá trị của x[i]

int main() {

int n, M;//khai báo n, M

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

cin >> n >> M;//nhập n, M

int m[n+1];//khai báo mảng n+1 phần tử, chứa giá trị các quả cân

for (int i = 1; i <= n; ++i) cin >> m[i];//nhập giá trị cho mảng m

int x[n+1];//khai báo mảng n+1 phần tử

stack<state> s;//khai báo stack s chứa các phần tử có kiểu state

//# sum of selected weights

int sum = 0;//khởi tạo sum = 0

s.push(state(1, -1));//đẩy phần tử đầu tiên vào stack

while (!s.empty()){//lặp tới khi s rỗng

state &top = s.top();//top = phần tử trên cùng của stack

if (top.i > n){//nếu xét hết tất cả quả cân

if (sum == M){//nếu sum = M

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

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

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

}//PHAN HOANG NAM - 20215434

cout << "=" << M;

exit(0);//dừng chương trình

}

s.pop();//xóa phần tử trên cùng của stack

continue;//chuyển sang vòng lặp tiếp theo

}

//# Khử đệ quy

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

# YOUR CODE HERE #\*/

if(top.j > -1){//nếu x[i]>-1 (backtrack)

sum = sum - m[top.i] \* x[top.i];//đặt lại sum cho lần lặp tiếp theo

}//PHAN HOANG NAM - 20215434

if(top.j > 1){//nếu x[i]>1

s.pop();//xóa phần tử trên cùng của stack

continue;//chuyển sang vòng lặp tiếp theo

}

x[top.i] = top.j;//gán x[i] = {-1, 0, 1} để thử

sum = sum + m[top.i] \* x[top.i];//tính sum

s.push(state(top.i+1, -1));//thử quả cân tiếp theo

top.j = top.j + 1;//tăng x[i] lên 1

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

}

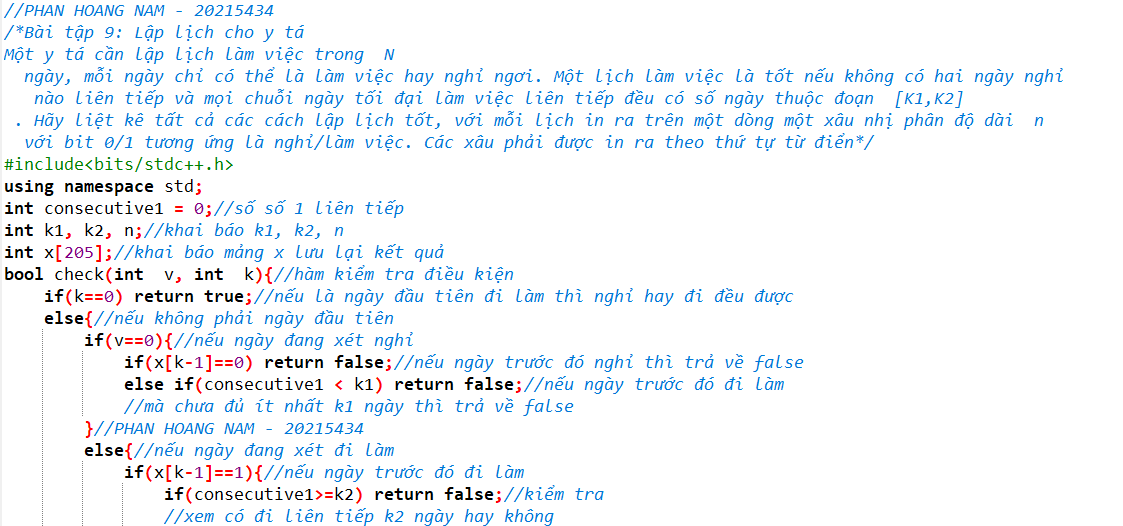
cout << -1;//nếu không có cách cân thì in ra -1

//PHAN HOANG NAM - 20215434

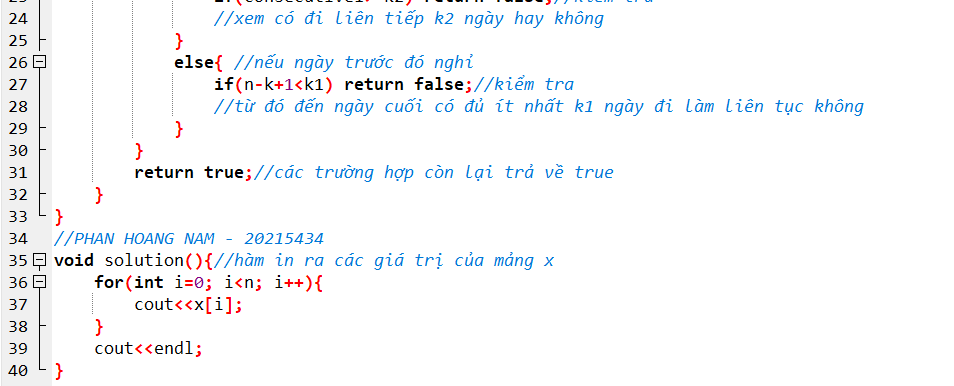
return 0;

}

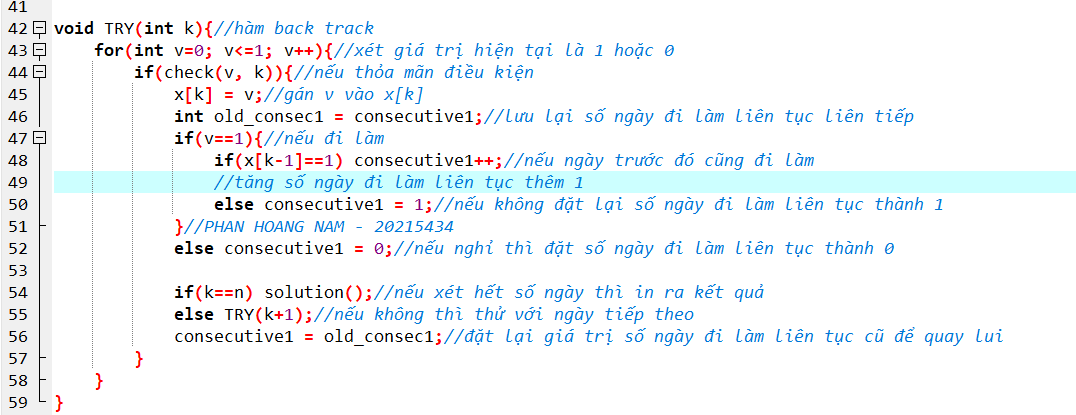
## Bài tập 9: Lập lịch cho y tá. Một y tá cần lập lịch làm việc trong N ngày, mỗi ngày chỉ có thể là làm việc hay nghỉ ngơi. Một lịch làm việc là tốt nếu không có hai ngày nghỉ nào liên tiếp và mọi chuỗi ngày tối đại làm việc liên tiếp đều có số ngày thuộc đoạn [K1,K2]. Hãy liệt kê tất cả các cách lập lịch tốt, với mỗi lịch in ra trên một dòng một xâu nhị phân độ dài N với bit 0/1 tương ứng là nghỉ/làm việc. Các xâu phải được in ra theo thứ tự từ điển



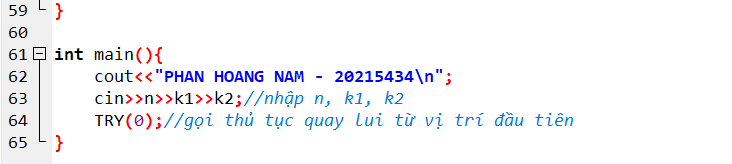
Hình . Code bài 3.9 hình 1



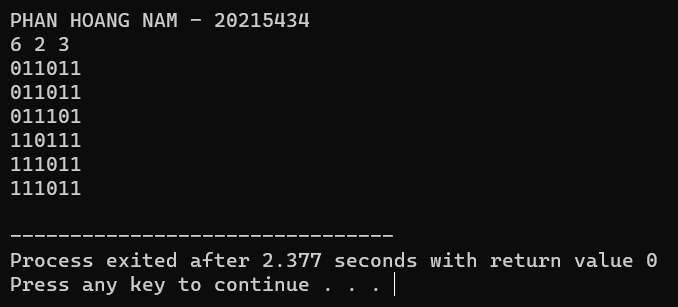
Hình . Code bài 3.9 hình 2



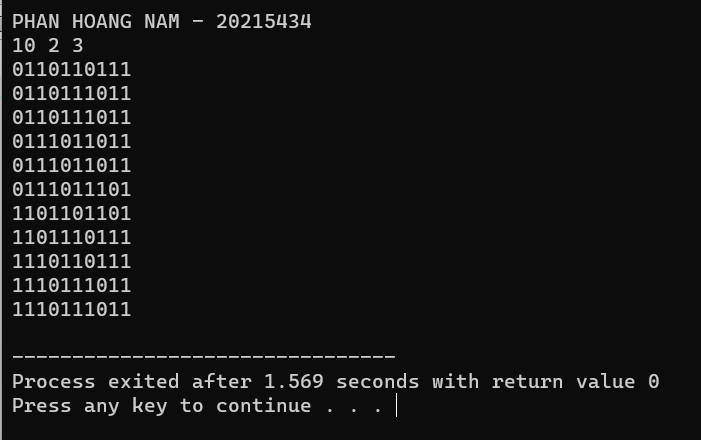
Hình . Code bài 3.9 hình 3



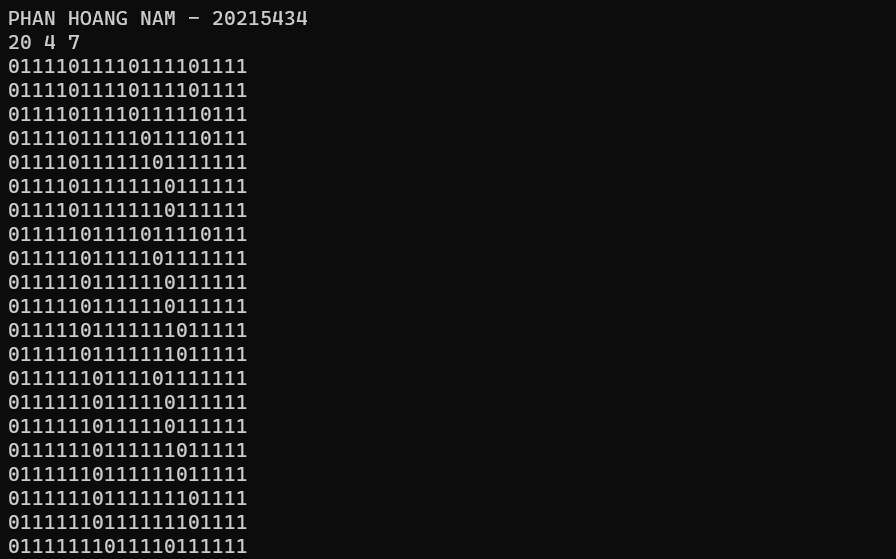
Hình . Code bài 3.9 hình 4



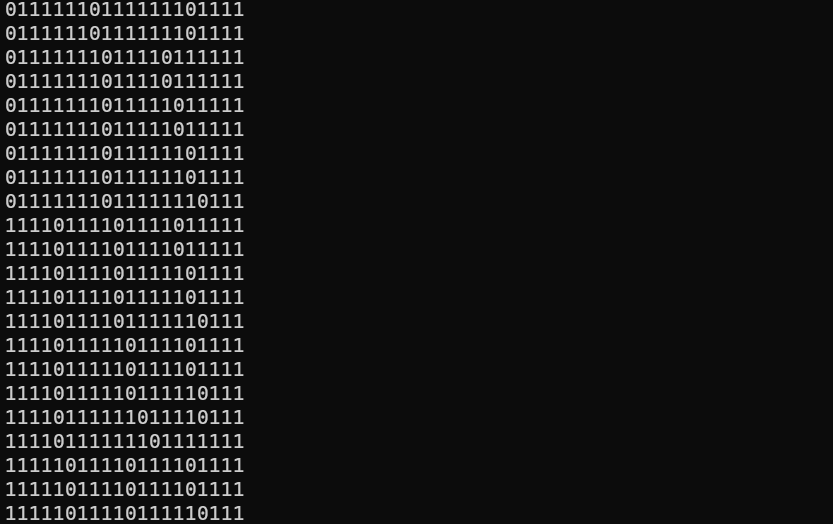
Hình . Output testcase 1 bài 3.9



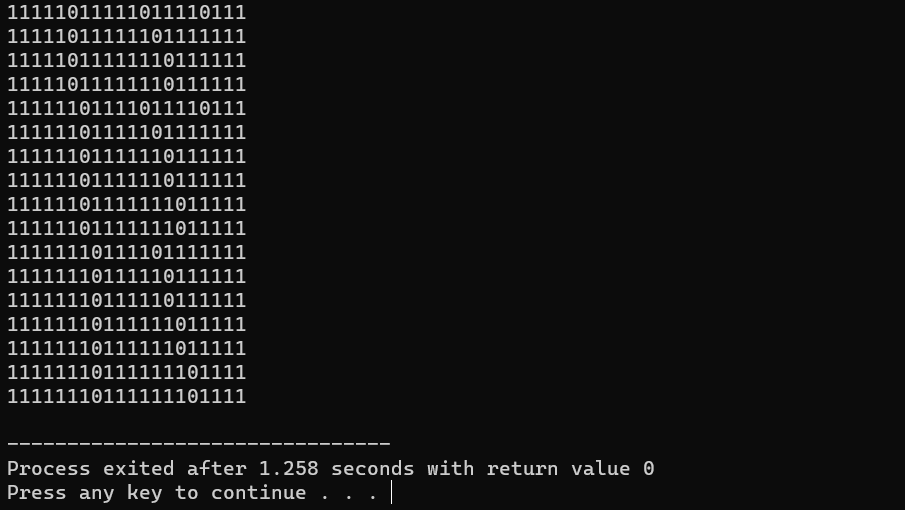
Hình . Output testcase 2 bài 3.9



Hình . Output testcase 3 bài 3.9



Hình . Output testcase 4 bài 3.9



Hình . Output testcase 5 bài 3.9

//PHAN HOANG NAM - 20215434

/\*Bài tập 9: Lập lịch cho y tá

Một y tá cần lập lịch làm việc trong N

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]

. Hãy liệt kê tất cả các cách lập lịch tốt, với mỗi lịch in ra trên một dòng một xâu nhị phân độ dài n

với bit 0/1 tương ứng là nghỉ/làm việc. Các xâu phải được in ra theo thứ tự từ điển\*/

#include<bits/stdc++.h>

using namespace std;

int consecutive1 = 0;//số số 1 liên tiếp

int k1, k2, n;//khai báo k1, k2, n

int x[205];//khai báo mảng x lưu lại kết quả

bool check(int v, int k){//hàm kiểm tra điều kiện

if(k==0) return true;//nếu là ngày đầu tiên đi làm thì nghỉ hay đi đều được

else{//nếu không phải ngày đầu tiên

if(v==0){//nếu ngày đang xét nghỉ

if(x[k-1]==0) return false;//nếu ngày trước đó nghỉ thì trả về false

else if(consecutive1 < k1) return false;//nếu ngày trước đó đi làm

//mà chưa đủ ít nhất k1 ngày thì trả về false

}//PHAN HOANG NAM - 20215434

else{//nếu ngày đang xét đi làm

if(x[k-1]==1){//nếu ngày trước đó đi làm

if(consecutive1>=k2) return false;//kiểm tra

//xem có đi liên tiếp k2 ngày hay không

}

else{ //nếu ngày trước đó nghỉ

if(n-k+1<k1) return false;//kiểm tra

//từ đó đến ngày cuối có đủ ít nhất k1 ngày đi làm liên tục không

}

}

return true;//các trường hợp còn lại trả về true

}

}

//PHAN HOANG NAM - 20215434

void solution(){//hàm in ra các giá trị của mảng x

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

cout<<x[i];

}

cout<<endl;

}

void TRY(int k){//hàm back track

for(int v=0; v<=1; v++){//xét giá trị hiện tại là 1 hoặc 0

if(check(v, k)){//nếu thỏa mãn điều kiện

x[k] = v;//gán v vào x[k]

int old\_consec1 = consecutive1;//lưu lại số ngày đi làm liên tục liên tiếp

if(v==1){//nếu đi làm

if(x[k-1]==1) consecutive1++;//nếu ngày trước đó cũng đi làm

//tăng số ngày đi làm liên tục thêm 1

else consecutive1 = 1;//nếu không đặt lại số ngày đi làm liên tục thành 1

}//PHAN HOANG NAM - 20215434

else consecutive1 = 0;//nếu nghỉ thì đặt số ngày đi làm liên tục thành 0

if(k==n) solution();//nếu xét hết số ngày thì in ra kết quả

else TRY(k+1);//nếu không thì thử với ngày tiếp theo

consecutive1 = old\_consec1;//đặt lại giá trị số ngày đi làm liên tục cũ để quay lui

}

}

}

int main(){

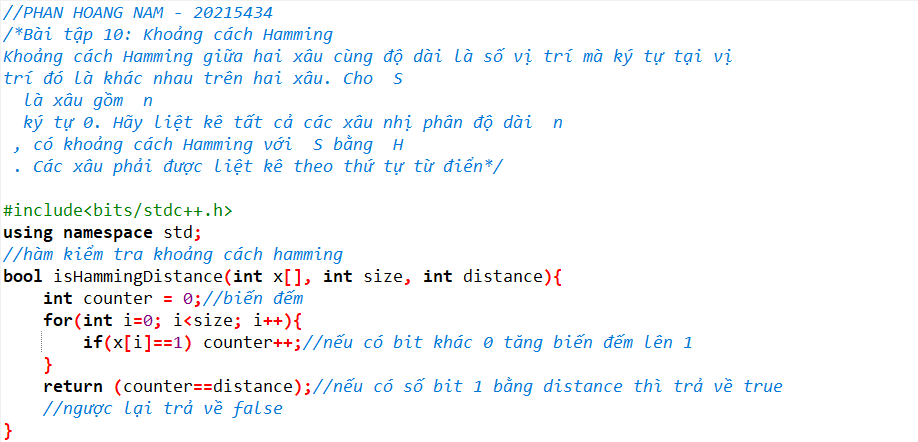
cout<<"PHAN HOANG NAM - 20215434\n";

cin>>n>>k1>>k2;//nhập n, k1, k2

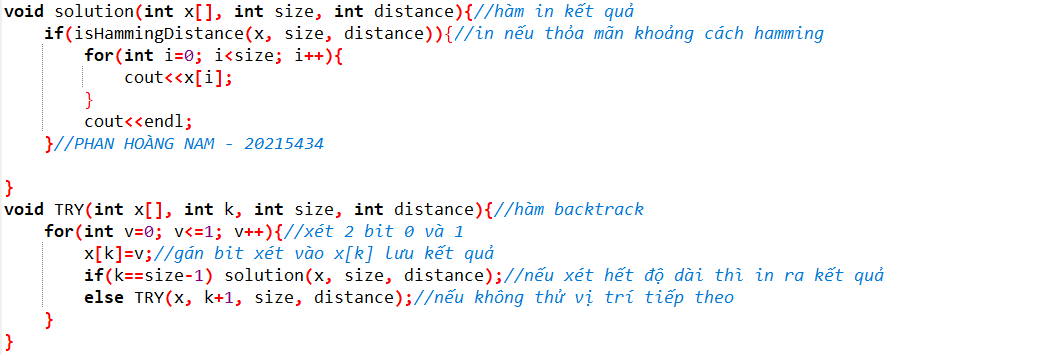
TRY(0);//gọi thủ tục quay lui từ vị trí đầu tiên

}

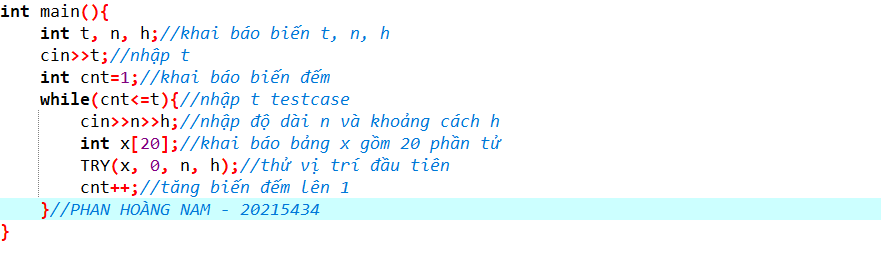
## Bài tập 10: Khoảng cách Hamming. Khoảng cách Hamming giữa hai xâu cùng độ dài là số vị trí mà ký tự tại vị trí đó là khác nhau trên hai xâu. Cho S là xâu gồm n ký tự 0. Hãy liệt kê tất cả các xâu nhị phân độ dài n, có khoảng cách Hamming với S bằng H. Các xâu phải được liệt kê theo thứ tự từ điển



Hình . Code bài 3.10 hình 1



Hình . Code bài 3.10 hình 2



Hình . Code bài 3.10 hình 3



Hình . Output testcase 1 bài 3.10 hình 1



Hình . Output testcase 1 bài 3.10 hình 2



Hình . Output testcase 1 bài 3.10 hình 3

//PHAN HOANG NAM - 20215434

/\*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 S

là xâu gồm n

ký tự 0. Hãy liệt kê tất cả các xâu nhị phân độ dài n

, có khoảng cách Hamming với S bằng H

. Các xâu phải được liệt kê theo thứ tự từ điển\*/

#include<bits/stdc++.h>

using namespace std;

//hàm kiểm tra khoảng cách hamming

bool isHammingDistance(int x[], int size, int distance){

int counter = 0;//biến đếm

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

if(x[i]==1) counter++;//nếu có bit khác 0 tăng biến đếm lên 1

}

return (counter==distance);//nếu có số bit 1 bằng distance thì trả về true

//ngược lại trả về false

}

void solution(int x[], int size, int distance){//hàm in kết quả

if(isHammingDistance(x, size, distance)){//in nếu thỏa mãn khoảng cách hamming

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

cout<<x[i];

}

cout<<endl;

}//PHAN HOÀNG NAM - 20215434

}

void TRY(int x[], int k, int size, int distance){//hàm backtrack

for(int v=0; v<=1; v++){//xét 2 bit 0 và 1

x[k]=v;//gán bit xét vào x[k] lưu kết quả

if(k==size-1) solution(x, size, distance);//nếu xét hết độ dài thì in ra kết quả

else TRY(x, k+1, size, distance);//nếu không thử vị trí tiếp theo

}

}

int main(){

int t, n, h;//khai báo biến t, n, h

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

cin>>t;//nhập t

int cnt=1;//khai báo biến đếm

while(cnt<=t){//nhập t testcase

cin>>n>>h;//nhập độ dài n và khoảng cách h

int x[20];//khai báo bảng x gồm 20 phần tử

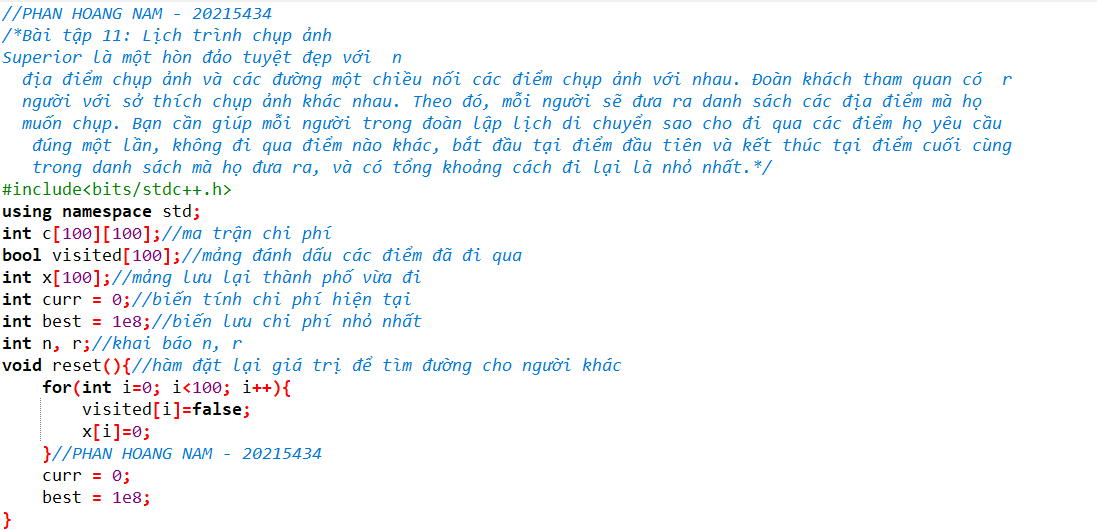
TRY(x, 0, n, h);//thử vị trí đầu tiên

cnt++;//tăng biến đếm lên 1

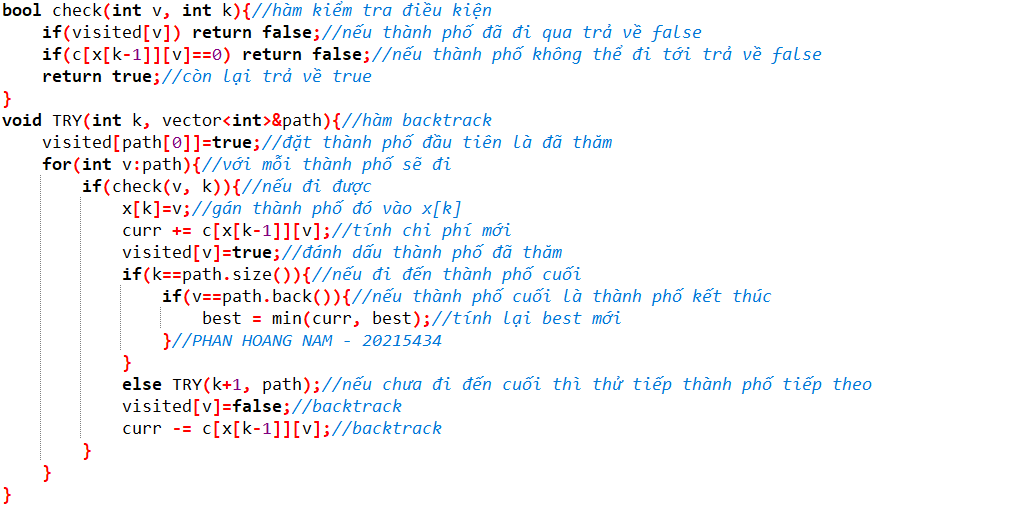
}//PHAN HOÀNG NAM - 20215434

}

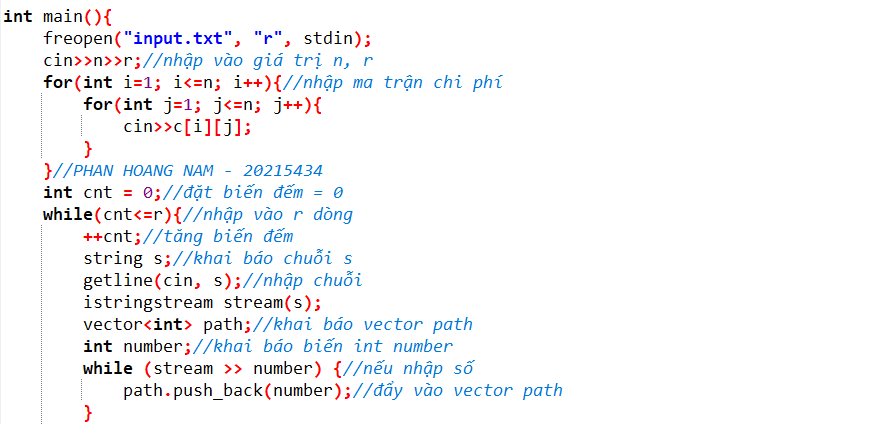
## Bài tập 11: Lịch trình chụp ảnh Superior là một hòn đảo tuyệt đẹp với n địa điểm chụp ảnh và các đường một chiều nối các điểm chụp ảnh với nhau. Đoàn khách tham quan có r người với sở thích chụp ảnh khác nhau. Theo đó, mỗi người sẽ đưa ra danh sách các địa điểm mà họ muốn chụp. Bạn cần giúp mỗi người 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.



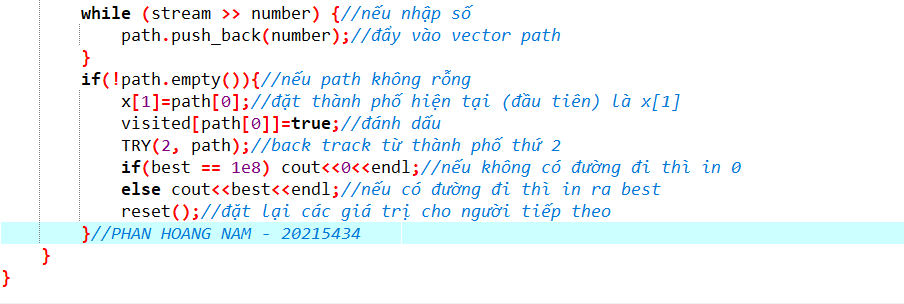
Hình . Code bài 3.11 hình 1



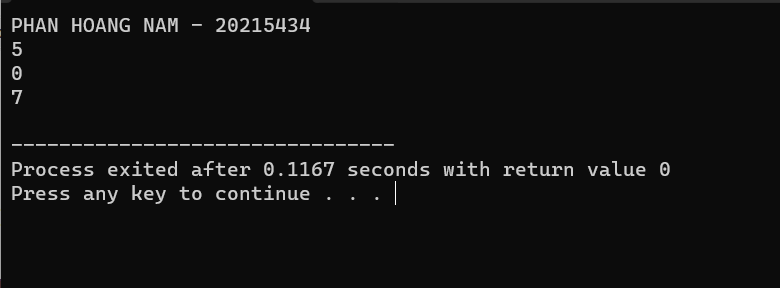
Hình . Code bài 3.11 hình 2



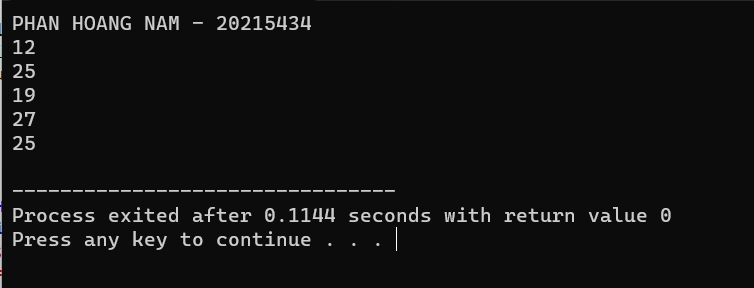
Hình . Code bài 3.11 hình 3



Hình . Code bài 3.11 hình 4



Hình . Output testcase 1 bài 3.11



Hình . Output testcase 2 bài 3.11

//PHAN HOANG NAM - 20215434

/\*Bài tập 11: Lịch trình chụp ảnh

Superior là một hòn đảo tuyệt đẹp với n

địa điểm chụp ảnh và các đường một chiều nối các điểm chụp ảnh với nhau. Đoàn khách tham quan có r

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.\*/

#include<bits/stdc++.h>

using namespace std;

int c[100][100];//ma trận chi phí

bool visited[100];//mảng đánh dấu các điểm đã đi qua

int x[100];//mảng lưu lại thành phố vừa đi

int curr = 0;//biến tính chi phí hiện tại

int best = 1e8;//biến lưu chi phí nhỏ nhất

int n, r;//khai báo n, r

void reset(){//hàm đặt lại giá trị để tìm đường cho người khác

for(int i=0; i<100; i++){

visited[i]=false;

x[i]=0;

}//PHAN HOANG NAM - 20215434

curr = 0;

best = 1e8;

}

bool check(int v, int k){//hàm kiểm tra điều kiện

if(visited[v]) return false;//nếu thành phố đã đi qua trả về false

if(c[x[k-1]][v]==0) return false;//nếu thành phố không thể đi tới trả về false

return true;//còn lại trả về true

}

void TRY(int k, vector<int>&path){//hàm backtrack

visited[path[0]]=true;//đặt thành phố đầu tiên là đã thăm

for(int v:path){//với mỗi thành phố sẽ đi

if(check(v, k)){//nếu đi được

x[k]=v;//gán thành phố đó vào x[k]

curr += c[x[k-1]][v];//tính chi phí mới

visited[v]=true;//đánh dấu thành phố đã thăm

if(k==path.size()){//nếu đi đến thành phố cuối

if(v==path.back()){//nếu thành phố cuối là thành phố kết thúc

best = min(curr, best);//tính lại best mới

}//PHAN HOANG NAM - 20215434

}

else TRY(k+1, path);//nếu chưa đi đến cuối thì thử tiếp thành phố tiếp theo

visited[v]=false;//backtrack

curr -= c[x[k-1]][v];//backtrack

}

}

}

int main(){

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

freopen("input.txt", "r", stdin);

cin>>n>>r;//nhập vào giá trị n, r

for(int i=1; i<=n; i++){//nhập ma trận chi phí

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

cin>>c[i][j];

}

}//PHAN HOANG NAM - 20215434

int cnt = 0;//đặt biến đếm = 0

while(cnt<=r){//nhập vào r dòng

++cnt;//tăng biến đếm

string s;//khai báo chuỗi s

getline(cin, s);//nhập chuỗi

istringstream stream(s);

vector<int> path;//khai báo vector path

int number;//khai báo biến int number

while (stream >> number) {//nếu nhập số

path.push\_back(number);//đẩy vào vector path

}

if(!path.empty()){//nếu path không rỗng

x[1]=path[0];//đặt thành phố hiện tại (đầu tiên) là x[1]

visited[path[0]]=true;//đánh dấu

TRY(2, path);//back track từ thành phố thứ 2

if(best == 1e8) cout<<0<<endl;//nếu không có đường đi thì in 0

else cout<<best<<endl;//nếu có đường đi thì in ra best

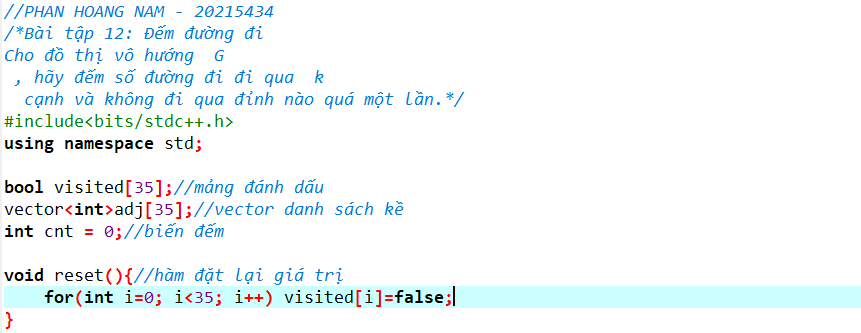
reset();//đặt lại các giá trị cho người tiếp theo

}//PHAN HOANG NAM - 20215434

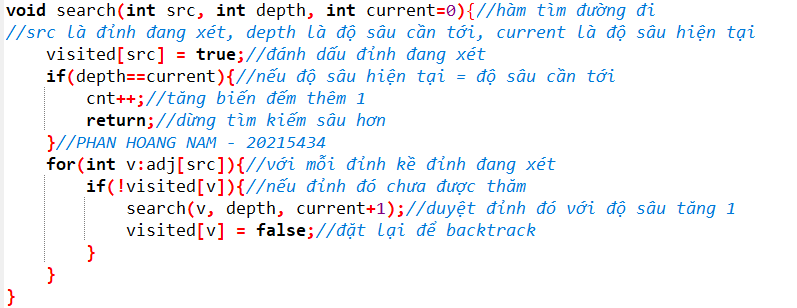
}

}

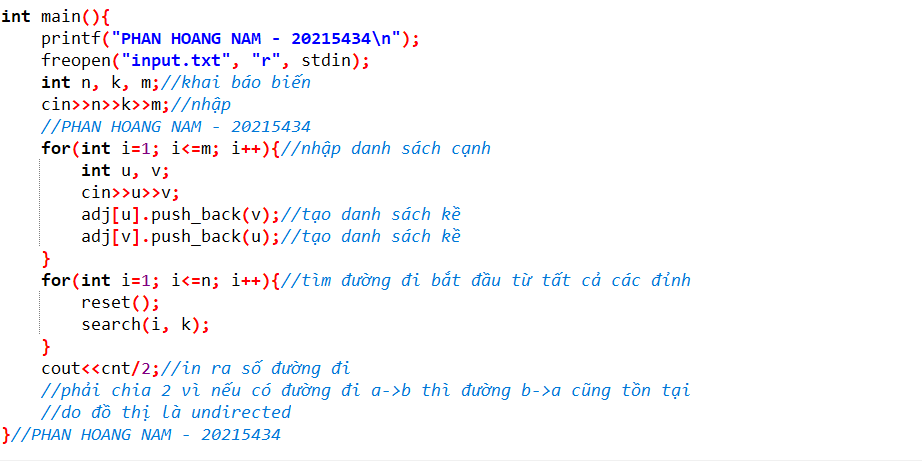
## Bài tập 12: Đếm đường đi Cho đồ thị vô hướng G, hãy đếm số đường đi đi qua k cạnh và không đi qua đỉnh nào quá một lần.



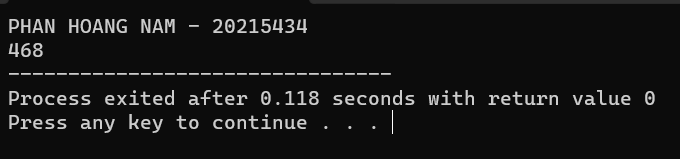
Hình . Code bài 3.12 hình 1



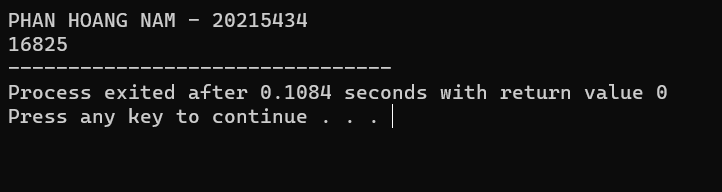
Hình . Code bài 3.12 hình 2



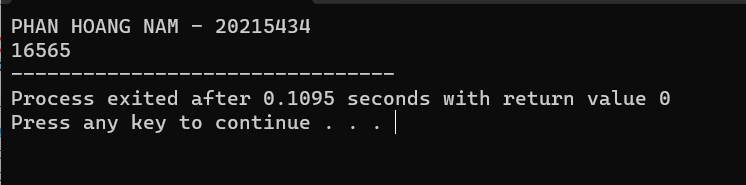
Hình . Code bài 3.12 hình 3



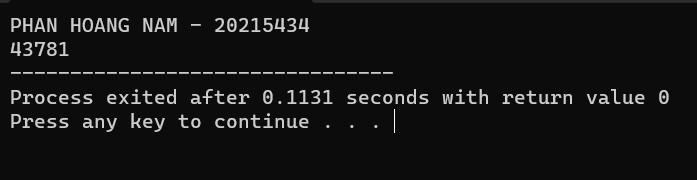
Hình . Output testcase 1 bài 3.12



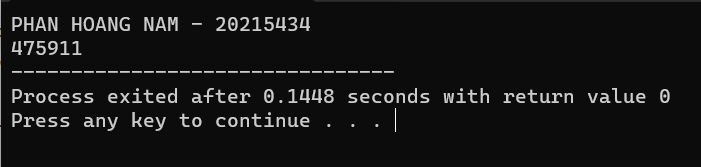
Hình . Output testcase 2 bài 3.12



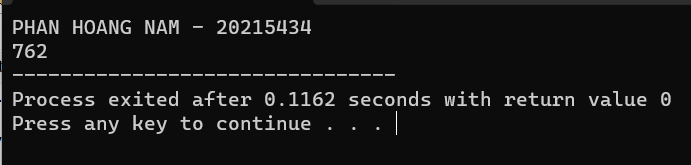
Hình . Output testcase 3 bài 3.12



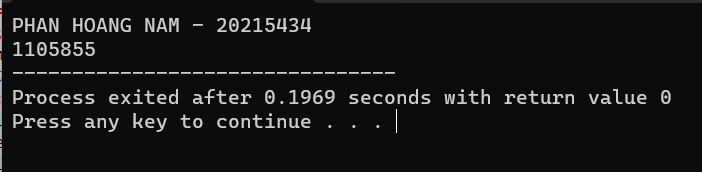
Hình . Output testcase 4 bài 3.12



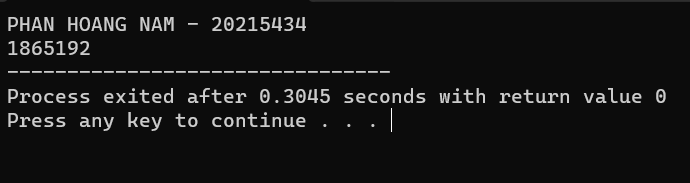
Hình . Output testcase 5 bài 3.12



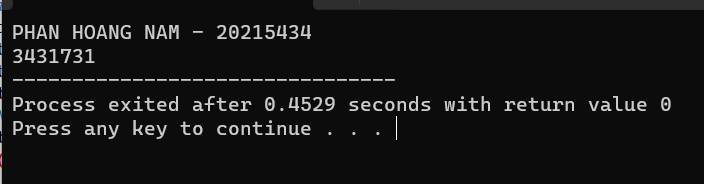
Hình . Output testcase 6 bài 3.12



Hình . Output testcase 7 bài 3.12



Hình . Output testcase 8 bài 3.12



Hình . Output testcase 9 bài 3.12

//PHAN HOANG NAM - 20215434

/\*Bài tập 12: Đếm đường đi

Cho đồ thị vô hướng G

, hãy đếm số đường đi đi qua k

cạnh và không đi qua đỉnh nào quá một lần.\*/

#include<bits/stdc++.h>

using namespace std;

bool visited[35];//mảng đánh dấu

vector<int>adj[35];//vector danh sách kề

int cnt = 0;//biến đếm

void reset(){//hàm đặt lại giá trị

for(int i=0; i<35; i++) visited[i]=false;

}

void search(int src, int depth, int current=0){//hàm tìm đường đi

//src là đỉnh đang xét, depth là độ sâu cần tới, current là độ sâu hiện tại

visited[src] = true;//đánh dấu đỉnh đang xét

if(depth==current){//nếu độ sâu hiện tại = độ sâu cần tới

cnt++;//tăng biến đếm thêm 1

return;//dừng tìm kiếm sâu hơn

}//PHAN HOANG NAM - 20215434

for(int v:adj[src]){//với mỗi đỉnh kề đỉnh đang xét

if(!visited[v]){//nếu đỉnh đó chưa được thăm

search(v, depth, current+1);//duyệt đỉnh đó với độ sâu tăng 1

visited[v] = false;//đặt lại để backtrack

}

}

}

int main(){

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

freopen("input.txt", "r", stdin);

int n, k, m;//khai báo biến

cin>>n>>k>>m;//nhập

//PHAN HOANG NAM - 20215434

for(int i=1; i<=m; i++){//nhập danh sách cạnh

int u, v;

cin>>u>>v;

adj[u].push\_back(v);//tạo danh sách kề

adj[v].push\_back(u);//tạo danh sách kề

}

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

search(i, k);

reset();

}

cout<<cnt/2;//in ra số đường đi

//phải chia 2 vì nếu có đường đi a->b thì đường b->a cũng tồn tại

//do đồ thị là undirected

}//PHAN HOANG NAM - 20215434