Câu 2: (2.5 điểm)
Viết chương trình đổi một số từ hệ thập lục phân sang hệ thập phân.
Dữ liệu vào: Nhập từ bàn phím một số duy nhất ở hệ thập lục phân.
Dữ liệu ra: Xuất ra màn hình một số nguyên duy nhất thể hiện kết quả sau khi đổi sang hệ thập phân.
Ví dụ:
INPUT
1234
OUTPUT
466

```
#include<iostream>
#include<string.h>
#include<math.h>
using namespace std;
int chuyenHe16Sang10(char S[]);
void nhap(char S[]);
void xuat(int kq);
int main()
{
    char S[100];
    nhap(S);
    int kq = chuyenHe16Sang10(S);
    xuat(kq);
    return 0;
}
int chuyenHe16Sang10(char S[])
{
    strupr(S);
    int len = strlen(S);
    int he16 = 0;
    int so = 0;
    int j = 0;
    for (int i = len - 1; i >= 0; i--)
        {
            if (S[i] >= 'A' && S[i] <= 'Z')
                so = S[i] - 55;
            else
                so = S[i] - 48;
            he16 = he16 + so * (pow(16,j));
            j++;
        }
    return he16;
}
void nhap(char S[])
{
    gets(S);
}
void xuat(int kq)
{
    cout<<kq;
}

```

ID: Chương 5 – Bài 1

Đổi tất cả các số thập phân từ 1 đến n sang hệ nhị phân

Dữ liệu đầu vào:

- Số nguyên dương N ($1 \leq N \leq 10^6$)

Dữ liệu đầu ra:

- N dòng đầu ra lần lượt là các số thập phân từ 1 đến N được chuyển qua hệ nhị phân

Ví dụ:

INPUT	OUTPUT
5	1 10 11 100 101

```cpp
include <iostream>
using namespace std;

// function to convert decimal to binary
void decToBinary(int n)
{
    // array to store binary number
    int binaryNum[1000];

    // counter for binary array
    int i = 0;
    while (n > 0) {

        // storing remainder in binary array
        binaryNum[i] = n % 2;
        n = n / 2;
        i++;
    }

    // printing binary array in reverse order
    for (int j = i - 1; j >= 0; j--)
        cout << binaryNum[j];
    cout<<endl;
}

// Driver program to test above function
int main()
{
    int n;
    cin>>n;
    for (int i = 1; i <= n; i++)
     decToBinary(i);
    return 0;
}

```

BÀI 1:

Bài 1: (2.5 điểm) Trong một kỳ thi, kết quả thi của mỗi thí sinh được lưu trong một dãy các ký hiệu có dạng "VXVVXVVVXX". Trong đó ký hiệu V cho biết câu trả lời tương ứng đúng và ký hiệu X cho biết câu trả lời tương ứng sai. Cách tính điểm cho mỗi bài thi (biết rằng số câu hỏi trong bài thi của thí sinh luôn nhỏ hơn 100) của thí sinh như sau:

Câu trả lời sai tính 0 (không) điểm. Trong dãy các câu trả lời đúng liên tiếp nhau, câu trả lời đúng đầu tiên tính 1 (một) điểm, các câu trả lời đúng tiếp theo được tính bằng **điểm của câu trả lời trước đó cộng thêm 1 (một)** điểm.

Như vậy với kết quả thi là "VXVVXVVVXX" sẽ được tính điểm như sau:

1 + 0 + 1 + 2 + 0 + 1 + 2 + 3 + 0 + 0 = 10 điểm.

Yêu cầu: Viết chương trình tính điểm cho mỗi thí sinh.

Dữ liệu vào: Một dòng duy nhất chứa kết quả thi của thí sinh. Giữa các ký hiệu V và X không có khoảng trắng.

Dữ liệu ra: Một số nguyên duy nhất cho biết điểm của bài thi đó.

Ví dụ:

Dữ liệu vào
VVXXVXXVVV
Dữ liệu ra
10

Dữ liệu vào
VXVXVXVXVXVXVX
Dữ liệu ra
7

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char S[100];
    gets(S);
    int N = strlen(S);
    int sum = 0;
    for(int i = 0; i < N; i++)
    {
        if(S[i] == 'V')
        {
            int temp = 0;
            for(i; i < N; i++)
            {
                if(S[i] == 'V')
                {
                    temp = temp +1;
                    sum += temp;
                }
                else break;
            }

        }
    }
    cout << sum;
}
```

BÀI 3:

Bài 3: (3 điểm) Cho một chuỗi ký tự S, hãy phân loại mỗi ký tự theo 4 kiểu sau: kiểu chữ thường, kiểu chữ in hoa, chữ số và kiểu "khác" (tức là các ký tự không thuộc ba loại trên) và cho biết mỗi loại có bao nhiêu ký tự.

Dữ liệu vào: Một chuỗi ký tự S có tối đa 200 ký tự.

Dữ liệu ra: Gồm 4 số nguyên a, b, c, d cho biết số ký tự của mỗi loại theo thứ tự lần lượt là số lượng của kiểu chữ thường (a), kiểu chữ in hoa (b), kiểu số (c) và kiểu khác (d), mỗi số cách nhau ít nhất một khoảng trắng

Ví dụ:

Dữ liệu vào
Hom nay la 13/07/2013

Dữ liệu ra
7 1 8 5

Giải thích ví dụ: *Có 7 ký tự thường là o,m,n,a,y,l.* *Có 1 ký tự in hoa là H*

Có 8 ký tự số là 1,3,0,7,2,0,1,3

Có 5 ký tự khác là 3 ký tự trắng và 2 ký tự /

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char S[200];
    gets(S);
    int len = strlen(S);
    int thuong = 0;
    int hoa = 0;
    int so = 0;
    int khac = 0;
    for (int i = 0; i < len;i++)
    {
        if ('0' <= S[i] && S[i] <='9')
            so++;
        else
            if ('a' <= S[i] && S[i]<='z')
                thuong++;
            else
                if ('A' <= S[i] && S[i] <='Z')
                    hoa++;
                else
                    khac++;
    }
    cout<<thuong<<" "<<hoa<<" "<<so<<" "<<khac;
}
```

BÀI 2:

Bài 2: (2 điểm) Cho một số nguyên n. Hỏi n có bao nhiêu bit 1 khi được biểu diễn ở dạng nhị phân 32 bit?

Dữ liệu vào: Một số nguyên n ($-10^9 \le n \le 10^9$).

Dữ liệu ra: Số lượng bit 1 của n.

Ví dụ:

Dữ liệu vào
17

Dữ liệu ra
2

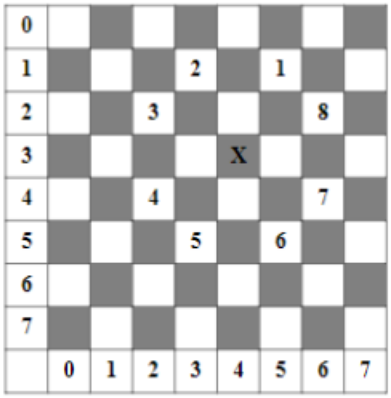
Giải thích ví dụ: *Dạng nhị phân của 17 là 00...0010001, do đó 17 có 2 bit 1.*

```cpp
#include<bits/stdc++.h>

using namespace std;

int main()
{
    int n;
    cin>>n;
    int k = 32;
    int cnt = 0;
    while (k--)
    {
        cnt+=n&0x1;
        n>>=1;
    }
    cout<<cnt;
    return 0;
}
```

Cho bàn cờ vua kích thước 8x8, các dòng, cột được đánh số từ 0 đến 7 và toạ độ quân mã. Biết rằng, nếu quân mã ở vị trí đánh dấu X, nó có thể di chuyển đến các vị trí được đánh số từ 1 đến 8 như trong hình.



Yêu cầu: Xác định xem trong một bước đi, quân mã có thể di chuyển từ vị trí (x1, y1) đến vị trí (x2, y2) cho trước hay không?

Dữ liệu vào: Lần lượt là bốn số nguyên x1, y1, x2, y2 ($0 \le x1, y1, x2, y2 < 8$) cho biết toạ độ của 2 vị trí trên bàn cờ lần lượt là (x1, y1) và (x2, y2).

Dữ liệu ra: Nếu đi không được thì xuất số 0. Ngược lại, xuất ra một số nguyên duy nhất có giá trị từ 1 đến 8 cho biết (x2, y2) thuộc về vị trí nào trong 8 vị trí di chuyển của quân mã từ vị trí (x1, y1).

Ví dụ:

Dữ liệu vào	Dữ liệu vào	Dữ liệu vào
3 4 2 6	3 5 6 7	2 0 4 1
Dữ liệu ra	**Dữ liệu ra**	**Dữ liệu ra**
8	0	6

```cpp
#include<iostream>
using namespace std;
    int X[8] = {-2,-2,-1,1,2,2,1,-1};
    int Y[8] = {1,-1,-2,-2,-1,1,2,2};
    int quanMa(int x1, int y1, int x2, int y2);
int main()
{
    int x1,x2,y1,y2;
    int n = 8;
    cin>>x1>>y1;
    cin>>x2>>y2;
    int kq = quanMa(x1,y1,x2,y2);
    cout<<kq;
}
int quanMa(int x1, int y1, int x2, int y2)
{
    for (int i = 0; i < 8;i++)
    {
        if (x1+X[i] == x2 && y1 + Y[i] == y2)
            return i+1;
    }
    return 0;
}
```

ID: Chương 4 – Bài 5

Tính C_N^K

Dữ liệu đầu vào:

- Một dòng duy nhất chứa hai số nguyên dương N, K ($1 \le K \le N \le 20$)

Dữ liệu đầu ra:

- Một dòng duy nhất là kết quả của bài toán

Ví dụ:

INPUT	OUTPUT
3 2	3

Giải thích:

- Ta có $C_3^2 = \frac{3!}{2!*1!} = \frac{1*2*3}{1*2*1} = 3$. Do đó kết quả $C_3^2 = 3$

```cpp

#include<iostream>
using namespace std;
int C(int k, int n) {
    if (k == 0 || k == n)
    return 1;
    if (k == 1)
    return n;
    return
    C(k - 1, n - 1) + C(k, n - 1);
}
int main(){
    int n, k;


    cin >> n>>k;
    cout <<C(k, n);

    return 0;
}
```

ID: Chương 4 – Bài 7

Tính tổng giá trị các số nguyên có trong một chuỗi ký tự

Dữ liệu đầu vào:

- Một dòng duy nhất chứa chuỗi ký tự X. Độ dài chuỗi ký tự này không quá 100000.

Dữ liệu đầu ra:

- Một dòng duy nhất là tổng các số có trong chuỗi kí tự này

Ví dụ:

INPUT	OUTPUT
2AS34ASDF342B	378

Giải thích:

- Ta có chuỗi 2AS34ASDF342B xuất hiện 3 con số lần lượt là 2, 34 và 342. Do đó tổng các giá trị số có trong chuỗi ký tự này là 378.

```cpp
#include <iostream>
#include <string.h>

using namespace std;
int main()
{
    char S[10000];
    gets(S);
    int sum = 0;
    int N = strlen(S);
    for(int i = 0; i < N; i++)
    {
        if('0' <= S[i] && S[i]  <= '9')
        {
            int temp = 0;
            for(i; i < N; i++)
            {
                if('0' <= S[i] && S[i]  <= '9')
                {
                    temp = temp * 10 + (S[i] - '0');
                }
                else break;
            }
            sum += temp;
        }
    }
    cout << sum;
}
```

ID: Chương 4 – Bài 10

Tính phần tử có tần suất xuất hiện nhiều nhất trong một mảng

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N ($1 \leq N \leq 1000$)
- Dòng thứ hai chứa N số nguyên $a_{i,j}$ ($|a_{i,j}| \leq 10^4$)

Dữ liệu đầu ra:

- Một dòng duy nhất là phần tử xuất hiện nhiều nhất trong mảng. Nếu có nhiều số có tần suất xuất hiện như nhau thì xuất ra số nhỏ nhất

Ví dụ:

INPUT	OUTPUT
5 1 2 1 2 1	1

Giải thích:

- Mảng có 5 phần tử trong đó có 3 số 1 và 2 số 2 do đó phần tử 1 có tần suất xuất hiện nhiều nhất nên kết quả của bài toán là 1

```cpp
#include<iostream>
using namespace std;
int tinhKQ(int n, int A[]);
bool timthay(int A[], int n, int x);
int demXH(int A[], int n, int x);
int main()
{
    int n, A[10000];
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> A[i];
    int kq = tinhKQ(n, A);
    cout << kq;
}
int tinhKQ(int n, int A[])
{
    int nB = 0;
    int C[1000];
    int B[1000];
    for (int i = 0; i < n; i++)
        if (timthay(B,nB,A[i]) == false)
            B[nB++] = A[i];
    for (int i = 0; i < nB; i++)
        C[i] = demXH(A,n,B[i]);
    int max = C[0];
    int vt = 0;
    for (int i = 0; i < nB; i++)
        if (max <= C[i])
        {
            max = C[i];
            vt = i;
        }
    int min = B[vt];
    for (int i = 0; i < nB;i++)
        if (C[i] == max && B[i] < min)
        {
            min = B[i];
        }
    return min;
}
bool timthay(int A[], int n, int x)
{
```

```cpp
}
bool timthay(int A[], int n, int x)
{
    for (int i = 0; i < n; i++)
        if (A[i] == x)
            return true;
    return false;
}
int demXH(int A[], int n, int x)
{
    int dem = 0;
    for (int i = 0; i < n; i++)
        if (A[i] == x)
            dem++;
    return dem;
}
```

ID: Chương 4 – Bài 11

Liệt kê tất cả các dãy nhị phân có độ dài N.

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N ($1 \leq N \leq 25$)

Dữ liệu đầu ra:

- Nhiều dòng trong đó mỗi dòng tượng trưng cho một chuỗi nhị phân có độ dài N. Các chuỗi được sắp xếp theo chiều tăng dần

Ví dụ:

INPUT	OUTPUT
3	000 001 010 011 100 101 110 111

Giải thích:

- Có tổng cộng $2^3 = 8$ chuỗi nhị phân có độ dài là 3 do đó cách sắp xếp trên là phù hợp với yêu cầu đề bài

```
#include<iostream>
#define SIZE 100
using namespace std;
int A[SIZE],n;

void lietKe(int k);
void xuat();

int main()
{

    cin>>n;
    lietKe(0);
}
void lietKe(int k)
{
    if (k == n)
        xuat();
    else
    {
        for (int i = 0; i <= 1; i++)
        {
            A[k] = i;
            lietKe(k+1);
        }
    }
}
void xuat()
{
    for (int i = 0; i < n; i++)
        cout<<A[i];
    cout<<endl;
}
```

ID: Chương 4 – Bài 15

Quốc gia X có N thành phố, được đánh số từ 0 đến $N-1$. Một người du lịch xuất phát từ một thành phố và muốn đi qua thăm tất cả các thành phố khác, mỗi thành phố đúng một lần rồi quay lại nơi xuất phát. Chi phí đi từ thành phố A đến thành phố B là $Cost[A][B]$ $(0 \leq A, B < N)$. Hãy tìm hành trình sao cho người du lịch để tổng chi phí theo hành trình này là ít nhất

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N $(1 \leq N \leq 10)$
- N dòng tiếp theo mỗi dòng chưa N số nguyên dương là giá trị đường đi từ đỉnh có chỉ số dòng tới đỉnh có chỉ số cột. $Cost[A][B]$ là giá trị đường đi từ đỉnh A đến đỉnh B. $(1 < Cost[A][B] \leq 10^4)$

Dữ liệu đầu ra:

- Một dòng duy nhất là giá trị đường đi nhỏ nhất

Ví dụ:

INPUT	OUTPUT
3 0 1 2 1 0 2 2 2 0	5

Giải thích:

- Đường đi từ đỉnh $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ có tổng đường đi là $1 + 2 + 2 = 5$.

```cpp
#include<iostream>
#define SIZE 20
using namespace std;
int A[SIZE],n,B[SIZE] = {0},dem;
int C[SIZE][SIZE];
int cptoiuu = INT_MAX;
int HT[SIZE];//hanh trinh

void lietKeHV(int k);
void nhap();
void xuat();
void lietKeHV(int k);
void tinhChiPhi();

int main()
{
    nhap();

    lietKeHV(0);
//  xuat();
    cout<<cptoiuu;
    return 0;
}
void nhap()
{
    cin>>n;
    for (int i = 0 ; i < n; i++)
        for (int j = 0; j < n; j++)
            cin>>C[i][j];
}
void lietKeHV(int k)
{
    if (k == n)
        tinhChiPhi();
    else
    {
        for (int i = 0; i < n; i++)
            if (B[i] == 0)
            {
                A[k] = i;
                B[i] = 1;
                lietKeHV(k+1);
                B[i] = 0;
            }
    }
}
void tinhChiPhi()
{
    int cp = 0;
    for (int i = 0; i < n-1; i++)
        cp = cp + C[A[i]][A[i+1]];
    cp = cp +C[A[n-1]][A[0]];
    if (cp < cptoiuu)
    {
        cptoiuu = cp;
        for (int j = 0; j < n ; j++)
            HT[j] = A[j];
    }
}
void xuat()
{

    for (int i = 0; i < n; i++)
        cout<<HT[i]<<" ";
    cout<<endl;
}
```

ID: Chương 5 – Bài 5

Viết hàm bitcount đếm số lượng bit 1 của một số nguyên dương N

Dữ liệu đầu vào:

- Số nguyên dương N ($1 \le N \le 10^9$)

Dữ liệu đầu ra:

- Một dòng duy nhất chứa số lượng bit 1 của số nguyên dương N

Ví dụ:

INPUT	OUTPUT
5	2

```cpp
#include<bits/stdc++.h>

using namespace std;

int main()
{
    int n;
    cin>>n;
    int k = 32;
    int cnt = 0;
    while (k--)
    {
        cnt+=n&0x1;
        n>>=1;
    }
    cout<<cnt;
    return 0;
}
```

ID: Chương 5 – Bài 8

Cho hàm $F(N)$ với N là số nguyên không âm được xác định như sau:

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(2n) = F(N) \\ F(2n+1) = F(N) + F(N+1) \end{cases}$$

Yêu cầu: Viết chương trình tính $F(N)$

Dữ liệu đầu vào:

- Số nguyên dương N ($1 \le N \le 10^6$)

Dữ liệu đầu ra:

- Một dòng duy nhất chứa kết quả $F(N)$ của bài toán

Ví dụ:

INPUT	OUTPUT
9	4

Giải thích:

0	1	2	3	4	5	6	7	8	9
0	1	1	2	1	3	2	3	1	4

Bên trên là biểu diễn giá trị theo số thứ tự của hàm $F(N)$ do đó kết quả của $F(9) = 4$

```cpp
#include<iostream>
using namespace std;
int tinhF(int n);
int main()
{
    int n;
    cin>>n;
    int kq = tinhF(n);
    cout<<kq;
    return 0;

}
int tinhF(int n)
{
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    else
    {
        if (n%2==0)
            return tinhF(n/2);
        else
            return tinhF((n-1)/2) + tinhF((n-1)/2 + 1);
    }
}
```

ID: Chương 5 – Bài 12

Bạn hãy liệt kê các hoán vị của tập N phần tử cho trước. Do có nhiều trường hợp đầu ra nên bạn hãy liệt kê các hoán vị theo thứ tự liên tiếp nhau từ nhỏ đến lớn.

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N là số phần tử của tập
- N phần tử tiếp theo là các giá trị A_i của mảng thoả điều kiện $|A_i| \leq 10^9$

Dữ liệu đầu ra:

- Nhiều dòng lần lượt là các hoán vị của N phần tử cho trước. Mỗi dòng chứa N phần tử

Ví dụ:

INPUT	OUTPUT
3 1 2 3	1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

Giải thích:

- Có tổng cộng 6 dãy hoán bị của 3 phần tử cho trước. Từ đó ta sắp xếp các dãy này theo thứ tự tăng dần ta được kết quả đầu ra

```cpp
#include<iostream>
using namespace std;
int A[100],B[100],check[100] ={0};
int n;
void lietkeHV(int k);
void xuat();
void latNguoc(int A[], int x, int y);
void doiCho(int &x, int &y);
void sapxep();
int main()
{

    cin>>n;
    for (int i = 0 ; i < n; i++)
        cin>>A[i];
    sapxep();
    lietkeHV(n);
    return 0;
}
void lietkeHV(int n)
{
//  for (int i = 0; i < n; i++)
//      B[i] = A[i];
    xuat();
    do
    {
        int k = n -2;
        while (k >= 0 && A[k] > A[k+1])
            k--;
        if (k < 0)
            break;
        int l = n -1;
        while (A[l] < A[k] )
            l--;
        doiCho(A[k],A[l]);
        int l = n -1;
        while (A[l] < A[k] )
            l--;
        doiCho(A[k],A[l]);
        latNguoc(A,k+1,n-1);
        xuat();
    }while (true);
}
void latNguoc(int A[], int x, int y)
{
    while(x < y)
    {
        doiCho(A[x],A[y]);
        x++;
        y--;
    }
}
void doiCho(int &x, int &y)
{
    int t = x;
    x = y;
    y =t;
}
void sapxep()
{
    for (int i = 0; i <n-1; i++)
        for (int j = i+1; j<n; j++)
            if (A[i] > A[j])
            {
                int tam = A[j];
                A[j] = A[i];
                A[i] = tam;
            }
}
void xuat()
{
    for (int i = 0; i < n; i++)
    {
        cout<<A[i]<<" ";
    }
    cout<<endl;
}
```

ID: Chương 6 – Bài 04

Viết hàm xoá các phần tử trùng nhau trong dãy, chỉ giữ lại một phần tử trong đó.

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N ($1 \le N \le 10^5$) biểu thị số lượng phần tử trong mảng
- Dòng tiếp theo chứa N số nguyên dương A_i ($|A_i| \le 10^9$) là các phần tử trong mảng

Dữ liệu đầu ra:

- Gồm một dòng duy nhất là chứa một số lượng số nguyên dương A_i thoả mãn yêu cầu đề bài. Chú ý: **Thứ tự đầu ra phải như thứ tự của dãy ban đầu.**

Ví dụ:

INPUT	OUTPUT
8 1 2 3 4 1 2 3 1	1 2 3 4

```cpp
#include<iostream>
using namespace std;
void tinhKQ(int n, int A[], int B[], int &nB);
bool timthay(int A[], int n, int x);
int demXH(int A[], int n, int x);
int main()
{
    int n, A[100000];
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> A[i];
    int nB = 0;
    int B[100000];
    tinhKQ(n, A,B,nB);
    for (int i = 0; i < nB; i++)
        cout<<B[i]<<" ";

}
void tinhKQ(int n, int A[], int B[], int &nB)
{

    for (int i = 0; i < n; i++)
        if (timthay(B,nB,A[i]) == false)
            B[nB++] = A[i];
}
bool timthay(int A[], int n, int x)
{
    for (int i = 0; i < n; i++)
        if (A[i] == x)
            return true;
    return false;
}

```

ID: Chương 6 – Bài 05

Bạn được cho hai ma trận vuông có kích thước NxN ($1 \le N \le 10^3$). Nhiệm vụ của bạn ở bài này là tính tích của hai ma trận vừa cho

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N ($1 \le N \le 10^3$)
- N dòng tiếp theo, mỗi dòng chứa N số nguyên dương $A_{i,j}$ ($|A_{i,j}| \le 10^3$) biểu diễn ma trận đầu tiên
- N dòng tiếp theo, mỗi dòng chứa N số nguyên dương $B_{i,j}$ ($|B_{i,j}| \le 10^3$) biểu diễn ma trận thứ hai

Dữ liệu đầu ra:

- N dòng tiếp theo chứa N số nguyên dương $C_{i,j}$ là kết quả của phép nhân hai ma trận A và B.

Ví dụ:

INPUT	OUTPUT
2 0 1 1 1 0 1 1 1	1 1 1 2

```cpp
#include<iostream>
using namespace std;
void tichMT(int A[][100], int B[][100], int n);
void nhap(int A[][100],int n);
int main()
{
    int n ;
    int A[100][100],B[100][100];
    cin>>n;
    nhap(A,n);
    nhap(B,n);
    tichMT(A,B,n);
    return 0;
}
void nhap(int A[][100],int n)
{
    for (int i = 0; i < n ; i++)
        for (int j = 0; j < n; j++)
            cin>>A[i][j];
}
void tichMT(int A[][100], int B[][100], int n)
{
    int sum = 0;
    int C[100][100];
    for (int i = 0; i < n; i++)
    {

        for (int j = 0; j < n; j++)
            {
                sum = 0;
                for (int k = 0; k < n; k++)
                    sum += (A[i][k]*B[k][j]);
                C[i][j] = sum;
            }


    }
    for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
                cout<<C[i][j]<<" ";
            cout<<endl;
        }

}
```

ID: Chương 6 – Bài 10

Bạn được cho một chuỗi các ký tự bao gồm một hoặc nhiều từ. Mỗi từ được định nghĩa là một chuỗi các ký tự được phân biệt bằng một hay nhiều dấu ký tự trắng.

Yêu cầu: Nhiệm vụ của bạn là viết chương trình đảo ngược lại các từ trong chuỗi cho trước. Kết quả trả ra phải được **chuẩn hoá.**

Dữ liệu đầu vào:

- Dòng duy nhất chứa một chuỗi các kí tự bao gồm các kí tự tiếng Anh không viết Hoa và dấu khoảng trắng. Độ dài của chuỗi không quá 100000 ký tự.

Dữ liệu đầu ra:

- Dòng duy nhất chứa chuỗi kí tự gồm các từ được đảo ngược lại

Ví dụ:

INPUT	OUTPUT
tu do hanh phuc	ut od hnah cuph

Giải thích:

- Xâu vừa cho bao gồm 4 từ bao gồm "tu", "do", "hanh" và "phuc" sau đó ta tiến hành đảo ngược các ký tự lại bao gồm "ut", "od", "hnah" và "cuhp" và gộp lại được kết quả như trên

```cpp
#include <bits/stdc++.h>

using namespace std;
void chuanhoa(string& s);
void Lower(string& s);
void daonguocchuoi(string& s, int x, int y);
void daonguoctungtu(string& s);

int main()
{
    /*char s;
    cin >> s;
    cout << int(s);*/
    string s;
    getline(cin, s);
    Lower(s);
    chuanhoa(s);
    cout << s;
    return 0;
}

void Lower(string& s)
{
    for (int i = 0; i < s.length(); i ++)
        if (s[i] >= 'A' && s[i] <= 'Z')
            s[i] = s[i] + 32;
}

void chuanhoa(string& s)
{
    int i = 0;
    while (i < s.length())
    {
        if (s[i] == ' ')
        {
            i++;
            continue;
        }
        if (s[i] < 'a' || s[i] > 'z')
        {
            s.erase(s.begin()+i);
        }
        else
            i++;
    }
    while (s[0] == ' ')
        s.erase(s.begin()+0);
    while (s[s.length()-1] == ' ')
        s.erase(s.begin()+s.length()-1);
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == ' ' && s[i+1] == ' ')
            s.erase(s.begin()+i);
    }
    s = s + " ";
    daonguoctungtu(s);
}

void daonguoctungtu(string& s)
{
    int vt = 0;
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == ' ')
        {
            daonguocchuoi(s,vt,i-1);
            vt = i + 1;
        }
    }
}

void daonguocchuoi(string& s, int x, int y)
{
    while (x < y)
    {
        swap(s[x], s[y]);
        x++;
        y--;
    }
}
```

ID: Chương 6 – Bài 11

Cho N số nguyên dương $A_1, A_2, \ldots, A_n$ bao gồm các số thực.

Yêu cầu: Bạn hãy tìm trong dãy trên một dãy con $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ của dãy trên thoả mãn điều kiện

$$\begin{cases} a_{i_j} \le a_{i_{j+1}} \\ i_j \le i_{j+1} \\ \text{Không có dãy con nào thoả mãn hai tính chất trên mà có nhiều phần tử hơn} \end{cases}$$

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N ($0 < N < 10^4$)
- Dòng tiếp theo chứa N số nguyên dương A_i ($|A_i| < 10^4$) là các phần tử trong mảng

Dữ liệu đầu ra:

- Một dòng duy nhất chứa **kích thước** của dãy con tìm được.

Ví dụ:

INPUT	OUTPUT
5 1 3 2 3 4	4

Giải thích:

- Trong 5 số vừa cho ở trên ta có thể tạo ra một dãy số gồm 4 số thoả mãn bao gồm [1, 2, 3, 4] do đó đầu ra của ta là 4.

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main ()
{
    int n;
    cin>>n;
    long arr[10000];
    long F[10000];
    for (int i=1; i<=n; i++)
        cin>>arr[i];
    arr[0] = 0;
    F[0] = 0;
    for (int i=1; i<=n; i++)
    {
        F[i] = 1;
        for (int j=i-1; j>=1; j--)
        {
            if (arr[i]>=arr[j])
            {
                F[i]=max(F[i], F[j]+1);
            }
        }
    }

    long dmax = 1;
    for (int i=1; i<=n; i++)
        if (F[i]>=dmax)
            dmax = F[i];
    cout<<dmax;
    return 0;
}
```

ID: Chương 6 – Bài 12

Bạn được cung cấp hai xâu A và B bao gồm các ký tự in hoà hoặ in thường và không chứa các ký tự đặc biệt. Hai xâu có độ dài nằm trong khoảng 50 đến 100. Xâu C được gọi là "xâu con chung" của A và B nhận được từ A (hoặc B) bằng cách loại bỏ một số ký tự nào đó thuộc xâu. Ví dụ A = "AdksHKoiGAdksHKoiG" và B = "ADKSHKOIGADKSHKOIG" thì C = "AHK"

Yêu cầu: Bạn hãy tìm **xâu con chung lớn nhất** có thể thu được

Dữ liệu đầu vào:

- Dòng đầu tiên chứa chuỗi ký tự A
- Dòng tiếp theo chứa chuỗi ký tự B

Dữ liệu đầu ra:

- Một dòng duy nhất chứa **kích thước** của xâu con chung dài nhất tìm được.

Ví dụ:

INPUT	OUTPUT
AdksHKoiGAdksHKoiG ADKSHKOIGADKSHKOIG	8

Giải thích:

- Như đã giải thích ở trên thì xâu C dài nhất ta có thể tìm được là "AHKGAHKG" và có độ dài là 8. Do đó kết quả trả ra của ta là 8

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    char A[100],B[100];
    char L[100][100];
    gets(A);
    gets(B);
    int m =strlen(A);
    int n = strlen(B);
    for (int i = 0; i <= m; i++)
        L[i][0] = 0;
    for (int i = 0; i <= n; i++)
        L[0][i] = 0;
    for (int i =1; i <= m;i++)
        for (int j = 1; j <= n; j++)
        {
            if (A[i-1] == B[j-1])
                L[i][j] = L[i-1][j-1]+1;
            else
                L[i][j] = max(L[i-1][j],L[i][j-1]);
            }
    long dmax = 1;
    for (int i=1; i<=m; i++)
    for (int j = 1; j <=n;j++ )
        if (L[i][j]>=dmax)
            dmax = L[i][j];
    cout<<dmax;
}
```

ID: Chương 6 – Bài 15

Hai anh em Chip và Dale được bác Zone tặng cho N gói kẹo. Gói kẹo thứ i có A_i viên kẹo. Chip và Dale đang cãi nhau vì không biết nên chia như thế nào cho công bằng nhất cho cả hai. Vì thế Chip và Dale nhờ bác Zone chia những gói kẹo thành 2 phần sao cho chênh lệch số kẹo của 2 phần là nhỏ nhất có thể.

Yêu cầu: Bạn hãy tìm số kẹo chênh lệch trong hai cách chia

Dữ liệu đầu vào:

- Dòng đầu tiên chứa N là số lượng gói kẹo bác Zone cho ($0 < N \leq 50$)
- Dòng tiếp theo chứa N số nguyên A_i là số lượng viên kẹo trong gói kẹo thứ i ($0 < A_i \leq 10^3$)

Dữ liệu đầu ra:

- Một dòng duy nhất là số lượng kẹo chênh lệch trong hai cách chia.

Ví dụ:

INPUT	OUTPUT
5 10 20 30 40 50	10

Giải thích:

- Cách chia tối ưu nhất ta có thể chia được là Chip sẽ nhận những gói kẹo có số lượng kẹo là {10, 20, 40} còn Dale sẽ nhận những gói kẹo có số lượng kẹo là {30, 50} như vậy chênh lệch ít nhất thu được là $|(10 + 20 + 40) - (30 + 50)| = 10$

```cpp
#include<iostream>
using namespace std;
int main()
{
    int n,A[50],result;
    cin>>n;
    for (int i = 1; i <= n; i++)
        cin>>A[i];
    int t = 0;
    for (int i = 1; i <= n; i++)
        t+= A[i];
    int s = t/2;
    int L[100000];
    L[0] = 1;
    for (int i= 1; i<=t/2;i++)
        L[i] = 0;
    for (int i = 1; i <=n;i++)
        for (int j = s;j >= A[i];j--)
        {
            if (L[j] == 0&& L[j-A[i]] == 1)
                L[j] = 1;
        }
    for (int i = s; i>=0;i--)
        if (L[i] == 1)
        {
            result = i;
            break;
        }
    cout<<(t-2*result);
}
```

Bài 2: (2 điểm) **CHIANHOM**

Cho M là một mảng các số nguyên dương có n phần tử.

Yêu cầu: Phân bổ các phần tử của mảng M vào 2 nhóm A và B sao cho chênh lệch của tổng giá trị các phần tử trong mỗi nhóm là nhỏ nhất.

Dữ liệu vào:

- Dòng đầu tiên là số nguyên dương n ($0<n \leq 20$).
- Dòng tiếp theo là n số nguyên dương, mỗi số nhỏ hơn 1000 chính là n phần tử của mảng M.

Dữ liệu ra:

Một số nguyên không âm duy nhất cho biết độ lệch nhỏ nhất sau khi chia M thành 2 nhóm.

Ví dụ:	**Dữ liệu vào**
	7 3 1 5 8 2 4 10
	Dữ liệu ra
	1
Giải thích:	*Nhóm A gồm các số: 3, 5, 8, tổng là 16.* *Nhóm B gồm các số: 1, 2, 4, 10, tổng là 17.* *Độ lệch không âm của tổng là: 17-16 = 1.*

```cpp
#include<bits/stdc++.h>
//#include<limits.h>
using namespace std;
int n,A[100],F[100];
long long solve()
{
    int min = 1e9;
    for (int i = 0; i<(1<<(n));i++)
    {
        int temp = i;
        int sum1 =0;
        int sum2 =0;
        for (int j = 0; j < n; j++)
        {
            F[j] = temp%2;
            temp/=2;
        }
        for (int j = 0; j < n; j++)
        {
            if (F[j] == 0)
                sum1+=A[j];
            else
                sum2+=A[j];
        }
        if (min>abs(sum1-sum2))
        {
            min = abs(sum1-sum2);
        }

    }
    return min;
}
int main()
{
    cin>>n;
    for (int i = 0; i<n;i++)
        cin>>A[i];
    cout<<solve();
}
```

Bài 4: (3 điểm) **THOADUOC**

Cho ma trận A kích thước MxN (2<M,N<100) chứa các số nguyên dương nhỏ hơn 100000 và một số nguyên K (0<K<100000). Một điểm $X_{i,j}$ được gọi là "thỏa được" nếu độ lệch của điểm này so với trung bình cộng (TBC) của các điểm lân cận (chung cạnh, chung đỉnh) nhỏ hơn K.

Ghi chú: Một điểm có tối thiểu 3 lân cận (điểm ở các góc), có tối đa 8 lân cận (điểm không nằm trên biên)

Yêu cầu: Cho biết ma trận A có bao nhiêu điểm "thỏa được" theo định nghĩa trên.

Dữ liệu vào: Có cấu trúc như sau:

+ Dòng đầu tiên lần lượt là ba số nguyên dương M, N, K.
+ M dòng tiếp theo, mỗi dòng là N số nguyên dương (mỗi số cách nhau ít nhất một khoảng trắng) lần lượt là N phần tử của từng dòng tương ứng của ma trận.

Dữ liệu ra: Xuất ra một số nguyên duy nhất cho biết số lượng điểm "thỏa được".

Ví dụ:	**Dữ liệu vào**				*Giải thích: Có 5 điểm "thỏa được" tại các vị trí được in đậm trên ma trận.* *TBC của lân cận số 5 là (4+9+8+3+6)/5 = 6, chênh lệch so với 5 là 1 (nhỏ hơn 2).* *TBC của lân cận số 11 là (9+8+12)/3 = 9.667, chênh lệch so với 11 là 1.333 (nhỏ hơn 2).* *TBC của lân cận số 3 là (5 + 6 + 7 + 8 + 2 + 12 + 10 + 1)/8 = 6.375, chênh lệch so với 3 là 3.375 (lớn hơn 2 – không "thỏa được")*
	3	4	2		
	4	**5**	**6**	7	
	9	**8**	3	2	
	11	12	10	1	
	Dữ liệu ra				
	5				

```cpp
using namespace std;
long long thoaDuoc(int m, int n, int k, int A[][105]);
int main()
{
    int m,n,k,A[105][105];
    cin>>m>>n>>k;

        for (int i = 0; i < m ; i++)
            for (int j = 0; j < n; j++)
                cin>>A[i][j];

    long long kq = thoaDuoc(m,n,k,A);
    cout<<kq;
}
long long thoaDuoc(int m, int n, int k, int A[][105])
{
    long long kq = 0;
    if (2 < m && 2 < n && 100 > m && 100 > n)
    {
    for (int i = 0; i < m ; i++)
        for (int j = 0; j < n; j++)
            {
                int tong = 0;
                int dem = 0;
                for (int f = -1; f <= 1; f++)
                    for (int q = -1; q<=1;q++)
                    {
                        if (i + f >= 0 && i + f <m && j+q>=0 && j+q < n)
                        {
                            dem++;
                            tong += A[i+f][q+j];
                        }
                    }
                tong = tong - A[i][j];
                dem = dem-1;
//              float tbc = float(tong)/(dem-1);
//              float tam = abs(tbc - A[i][j]);
                if ( abs(tong - A[i][j]*dem) < k*dem)
                    kq++;
            }
    }
    else
        kq = 0;
    return kq;
}
```

```cpp
#include<bits/stdc++.h>

using namespace std;

int m, n, k;
int A[105][105];

long long solve(){
    int Count = 0;
    for (int i=1;i<=m;i++)
        for (int j=1;j<=n;j++){
            int Sum = 0; //tong gia tri xung quanh
            int cnt = 0; //dem bao nhieu gia tri xung quanh
            for (int ii = -1;ii<=1;ii++)
                for (int jj = -1;jj<=1;jj++) {
                    if (i+ii!=0 && j+jj!=0 && i+ii<=m && j+jj<=n){
                        cnt+=1;
                        Sum+=A[i+ii][j+jj];
                    }
                }
            Sum-=A[i][j]; // tru di vi tri hien tai
            cnt-=1;       // tru di vi tri hien tai
            /*
                abs(sum/cnt-a[i][j])<k => abs(sum-a[i][j]*cnt)<k*cnt
            */
            if (abs(Sum-A[i][j]*(cnt))<(k*(cnt))) {
                Count+=1;
            }
    }
    return Count;
}

int main() {
    cin>> m>>n>>k;
    for (int i=1;i<=m;i++)
        for (int j=1;j<=n;j++) cin>>A[i][j];
    cout<<solve();
}
```

Bài 1: *(2.5 điểm)* **FIBO**

Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1, các phần tử sau đó được thiết lập theo quy tắc mỗi phần tử luôn bằng tổng hai phần tử trước nó. Công thức truy hồi của dãy Fibonacci là:

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(n) = F(n-1) + F(n-2) \quad khi\ n \geq 2 \end{cases}$$

Yêu cầu: Tính giá trị $F(n)$ của dãy Fibonacci nói trên.

Dữ liệu vào: Một số nguyên duy nhất n ($0 \leq n \leq 100$).

Dữ liệu ra: Một số nguyên duy nhất cho biết giá trị của $F(n)$

Ví dụ:	**Dữ liệu vào**
	10
	Dữ liệu ra
	55
Giải thích:	*Ta có chuỗi Fibonacci sẽ được biểu diễn ra dưới dạng mảng như sau* [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...], *do đó F(10) = 55.*

```cpp
#include<bits/stdc++.h>

using namespace std;

string Sum(string a, string b){
    while (a.size()<b.size()) a = "0" + a;
    while (a.size()>b.size()) b = "0" + b;
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    string result = "";

    int du = 0;
    int tong = 0;
    for (int i=0;i<a.size();i++){
        tong = (a[i]-'0')+(b[i]-'0')+du;
        du = tong / 10;
        result += (char)('0'+tong%10);
    }
    if (du) result += "1";
    reverse(result.begin(), result.end());
    return result;
}

string solve(int n){
    if (n==0) return "0";
    else if (n==1) return "1";
    string a = "0";
    string b = "1";
    string c = "";
    for (int i=2;i<=n;i++) {
        c = Sum(a, b);
        a = b;
        b = c;
    }
    return c;
}

int main(){
    int n;
    cin>>n;
    cout<<solve(n);
}
```

Bài 3: (2.5 điểm) **MAX**

Cho hai số nguyên dương X và Y (mỗi số có tối đa 20 chữ số). Bằng cách xóa đi một số chữ số của X, ta nhận được số nguyên dương Z. Giả sử số nguyên dương Z này cũng nhận được từ Y bằng cách xóa một số chữ số của Y.

Yêu cầu: Tìm số nguyên Z lớn nhất thỏa mãn yêu cầu trên.

Dữ liệu vào: Gồm 2 dòng, mỗi dòng lần lượt là số nguyên X và Y.

Dữ liệu ra: Một dòng duy nhất là số Z tìm được. Trường hợp không có Z nào thỏa thì xuất ra -1.

Lưu ý: Trong kết quả cần xóa đi những chữ số 0 vô nghĩa ở bên trái.

Ví dụ:	**Dữ liệu vào**	**Dữ liệu vào**	**Dữ liệu vào**
	12345 435012	19012304 34012	3557884 62919
	Dữ liệu ra	**Dữ liệu ra**	**Dữ liệu ra**
	45	34	-1

```cpp
#include<bits/stdc++.h>

using namespace std;

string A, B;

bool check(string Z){
    int id1 = 0;
    int id2 = 0;
    for (int i=0;i<A.size();i++){
        if (A[i]==Z[id1]) id1+=1;
        if (id1>=Z.size()) break;
    }
    for (int i=0;i<B.size();i++){
        if (B[i]==Z[id2]) id2+=1;
        if (id2>=Z.size()) break;
    }
    return (id1==Z.size() && id1==id2);
}

string Max(string A, string B){
    while (A.size()>0 && A[0]=='0') A.erase(0, 1);
    while (B.size()>0 && B[0]=='0') B.erase(0, 1);

    if (A.size()>B.size()) return A;
    if (A.size()<B.size()) return B;
    return max(A, B);
}

void solve(){
    int lenA = A.size();
    int lenB = B.size();
    int minLen = min(lenA, lenB);
    string Result = "";
    string base = "";
    if (minLen == lenA) base = A;
    else base = B;

    for (int i=0;i<(1<<(minLen));i++){
        int temp = i;
        string Z = "";
        for (int j=0;j<minLen;j++){
            if (temp%2==1) Z += base[j];
            temp/=2;
        }
        if (check(Z)) Result = Max(Result, Z);
    }
    if (Result == "") cout<< -1;
    else cout<<Result;
}

int main()
{
    getline(cin, A);
    getline(cin, B);
    solve();
}
```

MEETING – SẮP GIỜ CUỘC HỌP

Trường hiện có n cuộc họp được đánh số từ 1 đến N để đăng ký mượn phòng họp hội thảo tại tầng 6 toà nhà trung tâm. Cuộc họp thứ i cần được bắt đầu ngay sau thời điểm S_i và kết thúc tại thời điểm F_i. Giả sử bạn là người sắp xếp phòng họp, bạn hãy bố trí phòng hội thảo phục vụ được nhiều nhất bao nhiêu cuộc họp, sao cho khoảng thời gian làm việc của hai cuộc họp bất kỳ là không giao nhau.

Dữ liệu đầu vào:

- Dòng đầu tiên chứa số nguyên dương N ($N \leq 10.000$)
- Dòng thứ i trong số N dòng tiếp theo chứa hai số nguyên dương S_i, F_i ($S_i < F_i \leq 32.000$) ($1 \leq i \leq N$)

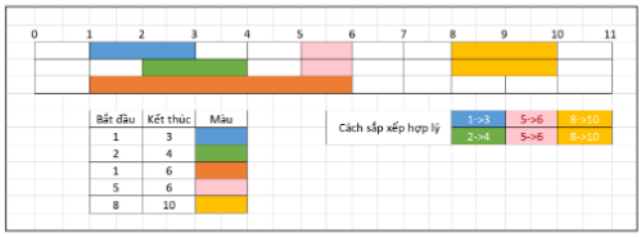
Dữ liệu đầu ra:

- Dòng duy nhất ghi số K là số các cuộc họp được chấp nhận phục vụ.

Ví dụ:

INPUT	OUTPUT
5 8 10 1 3 2 4 1 6 5 6	3

Giải thích:



- Ta chỉ có thể xếp được nhiều nhất 3 cuộc họp và có 2 cách xếp như giải thích trên

```cpp
#define second start
#define first finish

#include<iostream>
#include<algorithm>

using namespace std;

typedef pair<int, int> Time;

int n;
int Count;
Time a[10005];

void readData() {
    cin >> n;
    for (int i = 0; i < n; i++) cin >> a[i].start >> a[i].finish;
}

void solve() {
    sort(a, a + n);

    int startTime = -1;
    for (int i = 0; i < n; i++) {
        if (startTime <= a[i].start) {
            Count += 1;
            startTime = a[i].finish;
        }
    }
    cout << Count;
}

int main() {
    readData();
    solve();
}
```

RPS - OẲN TÙ TÌ

Có n người tham gia một giải đấu oẳn tù tì với số báo danh từ 1 tới n, giải đấu diễn ra với $n * \frac{n-1}{2}$ cặp đấu với mỗi người chơi đấu với chính xác n-1 người chơi khác, thể thức của trò chơi là mỗi người chọn sẽ chọn 1 trong 3 kí hiệu như kéo, búa và bao, với luật là kéo thắng bao, búa thắng kéo, bao thắng búa, 2 người chơi chọn kí hiệu giống nhau sẽ hòa. Với số điểm khi thắng cho 1 người chơi là 3, khi hòa là 1 và thua thì người chơi sẽ không có điểm nào. Bạn được cung cấp một danh sách chi tiết của từng vòng đấu, nhiệm vụ của bạn là sắp xếp số báo danh của n người chơi theo thứ tự số điểm giảm dần, trong trường hợp số điểm bằng nhau thì sẽ sắp xếp theo người chơi nào có số ván thắng nhiều hơn sẽ đứng trước, trong trường hợp tiếp tục bằng nhau thì người chơi nào có số báo danh nhỏ hơn sẽ đứng trước.

Dữ liệu đầu vào:

- Dòng đầu tiên gồm 1 số nguyên n ($2 \leq n \leq 100$) là số lượng người chơi.
- $\frac{n(n-1)}{2}$ dòng tiếp theo lần lượt chứa 4 số nguyên f, s, a, b với ($1 \leq f, s \leq n$) và ($1 \leq a, b \leq 3$) với f và s lần lượt là số báo danh của người chơi 1 và người chơi 2 trong cặp đấu này và ($f \neq s$) , a và b lần lượt là kí hiệu của người chơi 1 và người chơi 2 ra trong cặp đấu này, với quy định (1 = kéo, 2 = búa, 3 = bao) . Đầu vào đảm vào với mỗi số báo danh xuất hiện chính xác n-1 lần.

Dữ liệu đầu ra:

- Một danh sách là hoán vị của dãy {1,2,...,n} là yêu cầu của bài toán.

Ví dụ

INPUT	OUTPUT
3 1 2 1 3 1 3 2 2 3 2 3 2	1 3 2

Giải thích:

- Người số 1 thắng người số 2 và hoà người số 3 nên được 4 điểm
- Người số 2 thua người số 1 và 3 nên không có điểm
- Người số 3 hoà người số 1 và thắng người số 2 nên được 4 điểm

Do người 1 và người 3 bằng điểm, bằng số trận thắng nên người 1 xếp trước người 3 (do thứ tự nhỏ hơn)

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

struct PLAYER
{
    int id;
    int win = 0;
    int point = 0;
};

bool cmp(PLAYER a, PLAYER b)
{
    if (a.point > b.point)
        return true;
    else if (a.point == b.point) {
        if (a.win > b.win)
            return true;
        else if (a.win == b.win) {
            return a.id < b.id;
        }
    }

    return false;
}

int main()
{
    int n;
    cin >> n;

    PLAYER p[n+1];
    for (int i = 1; i <= n; ++i) {
        p[i].id = i;
        p[i].win = 0;
        p[i].point = 0;
    }

    for (int i = 1; i <= n*(n-1)/2; ++i) {
        int a, b, ca, cb;
        cin >> a >> b >> ca >> cb;
        if ((ca == 1 && cb == 3))
            ca = 4;
        if ((ca == 3 && cb == 1))
            cb = 4;

        if (ca == cb) {
            p[a].point++;
            p[a].win++;
            p[b].point++;
            p[b].win++;
        }
        else if (ca > cb) {
            p[a].point += 3;
            p[a].win++;
        }
        else {
            p[b].point += 3;
            p[b].win++;
        }
    }

    sort(p+1, p+n+1, cmp);

    for (int i = 1; i <= n; ++i)
        cout << p[i].id << ' ';
    return 0;
}
```

- Đặt $S(N) = 1 + (1^2 + 2^2) + \ldots + (1^2 + 2^2 + \ldots + n^2)$
- Nhận xét rằng mỗi phần tử của trong phép toán trên sẽ có dạng $F(X) = \sum_{K=1}^{X} k^2$ **(1)** với giá trị X nằm trong khoảng từ $1 \rightarrow N$.
- Sau khi có được nhận xét trên ta sẽ thu gọn phép toán lại được thành

$$S(N) = \sum_{X=1}^{N} \left(\sum_{k=1}^{X} K^2 \right) \qquad (2)$$

- Biến đổi giá trị (1) ta có:

$$F(X) = \sum_{K=1}^{X} k^2 = \frac{X * (X+1) * (2X+1)}{6} \qquad (3)$$

- Kết hợp (2) và (3) ta có được

$$S(N) = \sum_{X=1}^{N} \left(\frac{X * (X+1) * (2X+1)}{6} \right)$$

$$\rightarrow S(N) = \frac{1}{6} \sum_{X=1}^{N} (2X^3 + 3X^2 + X)$$

$$\rightarrow S(N) = \frac{1}{6} \left[2 \sum_{X=1}^{N} X^3 + 3 \sum_{X=1}^{N} X^2 + \sum_{X=1}^{N} X \right] \qquad (4)$$

- Lần lượt xử lý các giá trị nhỏ bên trong ta có được các cụm công thức sau:

$$\sum_{X=1}^{N} X^3 = \left(\sum_{X=1}^{N} X \right)^2 = \frac{X^2 (X+1)^2}{4}$$

$$\sum_{X=1}^{N} X^2 = \frac{X * (X+1) * (2X+1)}{6} \qquad (5)$$

$$\sum_{X=1}^{N} X = \frac{X * (X+1)}{2}$$

- Thay (5) vào (4) ta có được

$$S(N) = \frac{1}{6} \left[\frac{X^2 (X+1)^2}{2} + \frac{X(X+1)(2X+1)}{2} + \frac{X(X+1)}{2} \right]$$

Bài EQUATION (1)

$$\rightarrow S(N) = \frac{1}{12} [X^2 (X+1)^2 + X(X+1)(2X+1) + X(X+1)]$$

$$\rightarrow S(N) = \frac{1}{12} [X(X+1)(X^2 + X + 2X + 1 + 1)]$$

$$\rightarrow S(N) = \frac{1}{12} [X(X+1)^2 (X+2)]$$

- Sau khi có công thức trên, nếu sử dụng phương pháp tính toán thông thường ta chỉ được khoảng 50% số test do giới hạn đề bài quá lớn. Để giải quyết vấn đề trên ta sẽ sử dụng **BigInt ở chương 2** để giải quyết bài toán này.

```cpp
#include<iostream>
#include<algorithm>
#include<string>
#define long long ll
using namespace std;
const int Max = 1001;

/*
    S(n) = 1/6 * S(2n^3+3n^2+n)
    S(n) = 1/6 * [(n^2*(n+1)^2)/2 + n*(n+1)*(2n+1)/2 + n(n+1)/2]
    S(n) = 1/6 * [n*(n+1)*(n^2+n+2n+1+1)/2]
    S(n) = 1/12 * n*(n+1)^2*(n+2)
*/

string sum2Num(string a, string b) {
    if (a.size() < b.size()) swap(a, b);
    b.insert(0, a.size() - b.size(), '0');
    string ans = "";
    int count = 0;
    for (int i = a.size() - 1; i > -1; i--) {
        int tmp = (a[i] - '0') + (b[i] - '0') + count;
        ans += (tmp % 10 + '0');
        count = tmp / 10;
    }
    if (count) ans += (count + '0');
    reverse(ans.begin(), ans.end());
    return ans;
}

string mul2Num(string a, string b) {
    string ans = "";
    for (int i = a.size() - 1; i > -1; i--) {
        string res = "";
        res.insert(0, (a.size() - 1) - i, '0');
        int count = 0;
        for (int j = b.size() - 1; j > -1; j--) {
            int tmp = (b[j] - '0') * (a[i] - '0') + count;
            res += (tmp % 10 + '0');
            count = tmp / 10;
        }
        if (count) res += (count + '0');
        reverse(res.begin(), res.end());
        ans = sum2Num(ans, res);
    }
    return ans;
}
        ans = sum2Num(ans, res);
    }
    return ans;
}

int changeToNum(string s) {
    int value = 0;
    for (int i = 0; i < s.size(); i++)
        value = value * 10 + (s[i] - '0');
    return value;
}

string changeToString(int value) {
    string result = "";
    if (value == 0) return "0";
    while (value > 0) {
        result = (char)('0' + value % 10) + result;
        value /= 10;
    }
    return result;
}

string divide_12(string a) {
    if ((int)a.size() < 2 && changeToNum(a) < 12) return "0";
    string ans = "";
    int res = 0;
    for (int i = 0; i < a.size(); i++) {
        int tmp = res * 10 + (a[i] - '0');
        ans += changeToString(tmp / 12);
        res = tmp % 12;
    }
    while (ans[0] == '0') ans.erase(0, 1);
    return ans;
}

string solve(string n) {
    string num1 = sum2Num(n, "1");
    string num2 = sum2Num(n, "2");
    string result = mul2Num(mul2Num(num1, num1), mul2Num(n, num2));
    string ans = divide_12(result);
    return ans;
}

int main() {
    string n;
    cin >> n;
    cout << solve(n) << endl;
    return 0;
}
```

Bài RatAttack

```cpp
#include<iostream>

using namespace std;

int n, d;
int Max = -1;
int ResX, ResY;
int a[1050][1050];
int b[1050][1050];
const int MaxN = 1026;

int calc(int x1, int y1, int x2, int y2) {
    return b[x2][y2] - b[x2][y1 - 1] - b[x1 - 1][y2] + b[x1 - 1][y1 - 1];
}

void solve() {
    for (int i = 1; i <= MaxN; i++)
        for (int j = 1; j <= MaxN; j++)
            b[i][j] = b[i - 1][j] + b[i][j - 1] - b[i - 1][j - 1] + a[i][j];

    for (int i = 1; i <= MaxN - d; i++)
        for (int j = 1; j <= MaxN - d; j++) {
            int x1 = max(i - d, 1);
            int x2 = min(i + d, MaxN - 1);
            int y1 = max(j - d, 1);
            int y2 = min(j + d, MaxN - 1);
            int value = calc(x1, y1, x2, y2);
            if (value > Max) {
                Max = value;
                ResX = i - 1;
                ResY = j - 1;
            }
            else if (value == Max) {
                if ((ResX > i - 1) || (ResX == i - 1 && ResY > j - 1)) {
                    ResX = i - 1;
                    ResY = j - 1;
                }
            }
        }
    cout << ResX << ' ' << ResY << ' ' << Max << endl;
}

void cleanData() {
    for (int i = 1; i <= MaxN; i++)
        for (int j = 1; j <= MaxN; j++) {
```

0

```cpp
                }
            }
        }
    cout << ResX << ' ' << ResY << ' ' << Max << endl;
}

void cleanData() {
    for (int i = 1; i <= MaxN; i++)
        for (int j = 1; j <= MaxN; j++) {
            a[i][j] = 0;
            b[i][j] = 0;
        }
    Max = -1;
}

int main() {
    int query;
    cin >> query;
    while (query--) {
        cin >> d >> n;

        int x, y, value;
        for (int i = 0; i < n; i++) {
            cin >> x >> y >> value;
            a[x + 1][y + 1] = value;
        }

        solve();
        cleanData();
    }
}
```

ĐẾM TỪ

Cho một chuỗi ký tự (chiều dài tối đa 200 ký tự) chỉ chứa các ký tự từ a đến z, A đến Z và ký tự trắng.

Yêu cầu: Đếm xem trong chuỗi vừa nhập có bao nhiêu từ? Các từ được phân biệt bởi ít nhất một ký tự trắng.

Dữ liệu vào: Bàn phím gồm một dòng duy nhất chứa chuỗi cho trước (**cuối dòng không có ký tự xuống dòng (enter)**).

Dữ liệu ra: Xuất ra màn hình một số nguyên duy nhất cho biết số từ đếm được.

Ví dụ:

INPUT	OUTPUT
Cong nghe Thong tin	4

```cpp
#include<iostream>
#include<string.h>
using namespace std;
int demSoTuTrongChuoi(char S[]);
void nhap(char S[]);
void xuat(int kq);
int main()
{
    char S[100];
    nhap(S);
    int kq = demSoTuTrongChuoi(S);
    xuat(kq);
    return 0;
}
int demSoTuTrongChuoi(char S[])
{
    int dem = 0;
    int len = strlen(S);
    if (S[0] != 32)
        dem++;
    for (int i = 0; i < len - 1; i++)
        if (S[i] == 32 && S[i+1] != 32)
            dem++;
    return dem;
}
void nhap(char S[])
{
    gets(S);
}
void xuat(int kq)
{
    cout<<kq;
}
```

CHUẨN HÓA CHUỖI

Cho chuỗi ký tự S (chiều dài tối đa 2000 ký tự) chỉ chứa các ký tự từ a đến z, A đến Z và ký tự trắng.

Một chuỗi gọi là đã chuẩn hóa nếu không có khoảng trắng nào ở hai đầu và giữa các từ chỉ có duy nhất một khoảng trắng.

Yêu cầu: Xuất ra chuỗi S sau khi đã chuẩn hóa

Dữ liệu vào: Nhập từ thiết bị nhập chuẩn một dòng duy nhất là chuỗi S.

Dữ liệu ra: Xuất ra thiết bị xuất chuẩn một dòng duy nhất là chuỗi đã được chuẩn hóa của chuỗi S.

Ví dụ:

INPUT

Hom qua Qua noi Qua qua qua

OUTPUT

Hom qua Qua noi Qua qua qua

```cpp
using namespace std;
void xoaKhoangTrang(char S[], int &len);
void xoa(char S[], int &len, int k);
void nhap(char S[]);
void xuat(char S[], int len);
int main()
{
    char S[2000];
    nhap(S);
    int len = strlen(S);
    xoaKhoangTrang(S,len);
    xuat(S,len);
    return 0;
}
void xoaKhoangTrang(char S[], int &len)
{
    int i = 0;
    while(S[0] == ' ')
       xoa(S,len,0);
    while(S[len-1] == ' ')
       xoa(S,len,len-1);
    while(i < len)
       if(S[i] == ' ' && S[i+1] == ' ')
           xoa(S,len,i);
       else
           i++;
}
void xoa(char S[], int &len, int k)
{
    for (int i = k; i < len; i++)
       S[i] = S[i+1];
    len--;
}
void nhap(char S[])
{
    gets(S);
}
void xuat(char S[], int len)
{
    for (int i = 0; i < len; i++)
       cout<<S[i];
}
```

Bài 1: (2.5 điểm) Cho dãy số nguyên dương F_n được định nghĩa như sau:

$F_0 = 2$

$F_1 = 4$

$F_2 = 6$

$F_n = 2F_{n-3} + 4F_{n-2} + 6F_{n-1}$ (n>2)

Yêu cầu: Tìm chỉ số n lớn nhất thoả điều kiện $F_n \leq m$ ($1 < m < 10^8$) cho trước.

Dữ liệu vào: Một số nguyên dương duy nhất cho biết giá trị m.

Dữ liệu ra: Một số nguyên duy nhất cho biết giá trị n tìm được.

Ví dụ:	**Dữ liệu vào**	**Dữ liệu vào**	**Dữ liệu vào**
	2	10	60
	Dữ liệu ra	**Dữ liệu ra**	**Dữ liệu ra**
	0	2	3

```
#include<iostream>
using namespace std;
int tong(int n);
int tinh(int m)
{
    int i = 0;
    int n = 0;
    if (m >= 2 && m <4 )
        return 0;
    else
        if (m>=4 && m < 6)
            return 1;
        else
            if (m>=6 && m< 56)
                return 2;
            else
            {
                while (tong(n) <= m)
                {
                    i = n;
                    n++;

                }
                return i;
            }

}
int tong(int n)
{
    if (n == 0)
        return 2;
    if (n == 1)
        return 4;
    if (n == 2)
        return 6;
    else
        return (2*tong(n-3)+4*tong(n-2)+6*(tong(n-1)));
}
int main()
```

```
{
        int m;
    int i = 0;
    cin>>m;
    int kq = tinh(m);
    cout<<kq;
}
```

Bài 2: (2 điểm) Cho A là một mảng các số nguyên dương có n ($n < 25$) phần tử và một số nguyên dương k (k <100).

Yêu cầu: Chọn trong A các phần tử sao cho tổng các phần tử này chia hết cho k và đạt giá trị lớn nhất có thể.

Dữ liệu vào: Có cấu trúc như sau:

- Dòng đầu tiên lần lượt là hai số nguyên dương n và k.
- Dòng tiếp theo là n số nguyên dương nhỏ hơn 1.000.000 (mỗi số cách nhau ít nhất một khoảng trắng) lần lượt là n phần tử của mảng A.

Dữ liệu ra: Một số nguyên duy nhất cho biết tổng các phần tử chọn được.

Ví dụ:	**Dữ liệu vào**	**Dữ liệu ra**
	5 7 33 54 51 48 15	168

```
#include <iostream>
#include <math.h>

#define SIZE 100

using namespace std;

void nhap(int &n, int &k, int a[])
{
    cin >> n >> k;
    for (int i = 0; i < n; i++)
        cin >> a[i];
}

int tinhTong(int a[], int tapCon[], int n)
{
    int tong = 0;
    for (int i = 0; i < n; i++)
    {
        if (tapCon[i] == 1)
        {
            tong += a[i];
        }
    }
    return tong;
}

void sinhTapCon(int tapCon[], int n, int t, int a[], int KQ[], int &nKQ)
{
    if (t == n)
    {
        KQ[nKQ++] = tinhTong(a, tapCon, n);
    }
    else
```

```
    {
        for (int i = 0; i <= 1; i++)
        {
            tapCon[t] = i;
            sinhTapCon(tapCon, n, t + 1, a, KQ, nKQ);
        }
    }
}

int timKQ(int KQ[], int nKQ, int k)
{
    int chiaHetK[SIZE], n = 0;
    for (int i = 0; i < nKQ; i++)
    {
        if (KQ[i] % k == 0)
            chiaHetK[n++] = KQ[i];
    }
    int max = chiaHetK[0];
    for (int i = 1; i < n; i++)
    {
        if (chiaHetK[i] > max)
        {
            max = chiaHetK[i];
        }
    }
    return max;
}

int main()
{
    int n, k;
    int a[SIZE];
    int tapCon[SIZE];
    int KQ[SIZE], nKQ = 0;
    nhap(n, k, a);
    sinhTapCon(tapCon, n, 0, a, KQ, nKQ);
    int kq = timKQ(KQ, nKQ, k);
    cout << kq << endl;
    return 0;
}
```

NOTBB

Đề bài :

Cho số nguyên dương **n** (**n ≤ 20**) , hãy liệt kê tất cả các xâu độ dài **n** chỉ gồm 2 kí tự **'A'** hoặc **'B'** mà không có 2 kí tự **'B'** nào đứng cạnh nhau. Trong đó, kết quả được in theo thứ tự từ điển. Chuỗi **A** được coi là nhỏ hơn chuỗi **B** trong thứ tự từ điển nếu tồn tại một vị trí i mà **A[i] < B[i]** hoặc độ dài chuỗi **A** bé hơn **B** và **A** là một tiền tố của **B**.

Đầu vào :

- Một dòng duy nhất chứa số nguyên dương **n**.

Đầu ra :

- Một số dòng theo yêu cầu bài toán.

Ví dụ :

Đầu vào	Đầu ra
4	AAAA AAAB AABA ABAA ABAB BAAA BAAB BABA

```
#include<iostream>


#define SIZE 100

using namespace std;

void sinhNhiPhan(int n);
void print(int A[], int n);

int main()
{
    int n;
    cin>>n;
    sinhNhiPhan(n);
    return 0;
}
void sinhNhiPhan(int n)
{
    int A[SIZE] = {0};
    int i;
    do
    {
        i = n-1;
        bool flag = true;
        for (int k = 0; k<n-1; k++)
            if (A[k]==1 && A[k+1]==1)
```

```
            {
                flag=false;
                break;
            }

        if (flag==true)
            print(A, n);
        while (i>=0 && A[i]==1)
        {
            A[i] = 0;
            i--;
        }
        if (i>=0)
            A[i] = 1;
    } while (i>=0);
}
void print(int A[], int n)
{
    for (int i = 0; i<n; i++)
        if (A[i]==0)
            cout<<'A';
        else
            cout<<'B';
    cout<<endl;
}
```

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT

THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

NGÀNH CÔNG NGHỆ THÔNG TIN

ĐÁP ÁN CUỐI KỲ HỌC KỲ 2 NĂM HỌC 15-16

Môn: KỸ THUẬT LẬP TRÌNH

Mã môn học: PRTE240385

Đề số/Mã đề: **01** Đáp án có **5** trang.

1) **Hàm đọc dữ liệu từ tập tin (1.5đ):**

```
void docfile(char *filename, int M[][MAX], int &N)
{
        FILE *fp;
        fopen_s(&fp, filename, "rt");
        if (!fp) return;
        // đọc cấp của ma trận
        fscanf_s(fp, "%d", &N);
        // đọc nội dung ma trận
        for (int i = 0; i<N; i++)
                for (int j = 0; j<N; j++)
```

```
                        fscanf_s(fp, "%d", &M[i][j]);
        fclose(fp);
}
```

2) Hàm kiểm tra giá trị từ 1 đến N^2 (1đ): không so sánh với 1 trừ 0.5đ

```
int ktgiatri(int M[][MAX], int N)
{
        for (int i = 0; i < N; i++)
                for (int j = 0; j < N; j++)
                        if (M[i][j]<1 || M[i][j]>N*N)
                                return 0;
        return 1;
}
```

3) Hàm kiểm tra trùng nhau (1đ):

```
int kttrung(int M[][MAX], int N)
{
        for (int i = 0; i < N*N - 1; i++)
                for (int j = i + 1; j < N*N; j++)
                        if (M[i/N][i%N] == M[j/N][j%N])
                                return 1;
        return 0;
}
```

4) Hàm kiểm tra ma phương (1.5đ):

```
//tổng các phần tử dòng k (0.25đ)
int tongdong(int M[][MAX], int N, int k)
{
        int t = 0;
        for (int i = 0; i < N; i++)
                t += M[k][i];
        return t;
}
//tổng các phần tử cột k (0.25đ)
int tongcot(int M[][MAX], int N, int k)
{
        int t = 0;
        for (int i = 0; i < N; i++)
                t += M[i][k];
        return t;
}
//tổng chéo chính (0.25đ)
int tongcheo(int M[][MAX], int N)
{
        int t = 0;
        for (int i = 0; i < N; i++)
                t += M[i][i];
```

```
        return t;
}
//tổng chéo phụ (0.25đ)
int tongcheo2(int M[][MAX], int N)
{
        int t = 0;
        for (int i = 0; i < N; i++)
                t += M[i][N-1-i];
        return t;
}

int ktmaphuong(int M[][MAX], int N)      //0.5đ
{
        if (!ktgiatri(M, N)) return 0;
        if (kttrung(M, N)) return 0;
        int t = tongdong(M, N, 0);
        if (tongcheo(M, N) != t) return 0;
        if (tongcheo2(M, N) != t) return 0;
        for (int i = 0; i < N; i++)
        {
                if (tongdong(M, N, i) != t) return 0;
                if (tongcot(M, N, i) != t) return 0;
        }
        return 1;
}
```

5) Tìm đường đi tổng lớn nhất bằng phương pháp tham lam (2đ):
Không tìm max của dòng đầu chỉ được ½ số điểm.

```
int thamlam(int M[][MAX], int N, int path[])
{
        int max = M[0][0];
        int cur = 0;
        for (int i = 1; i < N; i++)
                if (max < M[0][i]) {
                        max = M[0][i];
                        cur = i;
                }
        int tong = M[0][cur];
        int v;
        path[0] = cur;
        for (int i = 1; i < N; i++)
        {
                max = 0;
                for (int j = -1; j <= 1; j++)
```

```
                        if (cur + j >= 0 && cur + j < N
                            && max < M[i][cur + j])
                        {
                                max = M[i][cur + j];
                                v = cur + j;
                        }
                tong += max;
                path[i] = v;
                cur = v;
        }
        return tong;
}
```

6) **Tạo mảng 1 chiều (1đ):**

```
void taomang1D(int M[][MAX], int N, int A[])
{
        for (int i = 0; i < N*N; i++)
                A[i] = M[i/N][i%N];
}
```

Tìm dãy con giảm dài nhất (2đ):

```
int qhdong(int a[], int n, int L[], int T[])
{
        //định nghĩa giá trị vô cùng
        int MAX_INT = (int)pow(2.0, 31) - 1;            //2^31 - 1
        //dời mảng ra sau 1 vị trí để chèn -VC vào
        memmove(a + 1, a, n*sizeof(int));
        //chèn 2 giá trị sentinel vào
        a[0] = MAX_INT;
        a[n + 1] = -MAX_INT;
        L[n + 1] = 1;                   //cơ sở
        for (int i = n; i >= 0; i--)    //chạy ngược lên đầu mảng
        {
                //Tìm vị trí có L[j] lớn nhất
                int jmax = n + 1;
                for (int j = i + 1; j <= n + 1; j++)
                        if (a[j] < a[i] && L[j] > L[jmax])
                                jmax = j;               //lưu lại vị trí lớn nhất
                //Tính L[i] và T[i]
                L[i] = L[jmax] + 1;
                T[i] = jmax;
        }
        return L[0] - 2;        //loại 2 sentinel ra
}
```

Nội dung bổ sung:

Hàm in ma trận:

```
void inmt(int M[][MAX], int N)
{
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++) {
			cout.width(4);
			cout << M[i][j];
		}
		cout << endl;
	}
}
```

Hàm main():

```
int main()
{
	int m[MAX][MAX];
	int n;
	//Câu 1
	docfile("F:/MAPHUONG.TXT", m, n);
	inmt(m, n);
	//Câu 2,3,4
	if (ktmaphuong(m, n))
		cout << "Day la ma tran ky ao!\n";
	else
		cout << "Day khong phai ma tran ky ao!\n";
	//Câu 5
	int path[MAX];
	int t = thamlam(m, n, path);
	cout << "Tong duong di: " << t << endl;
	cout << "Duong di: ";
	for (int i = 0; i < n; i++)
		cout << path[i] << "-->";
	cout << "end" << endl;
	//Câu 6
	int A[MAX*MAX + 2], L[MAX*MAX+2], T[MAX*MAX+2];
	taomang1D(m, n, A);
	for (int i = 0; i < n*n; i++)
		cout << A[i] << "  ";
	cout << endl;
	t = qhdong(A, n*n, L, T);
	cout << "Do dai mang con: " << t << endl;
	cout << "Mang con: ";
	int i = T[0];
	while (i != n*n + 1) {
```

```
            cout << A[i] << "  ";
            i = T[i];
        }
        cout << endl;
    return 0;
}
```

DDEE 2016

Ma trận "hình thoi" là một ma trận vuông có tính chất như sau:

- Cấp N của ma trận là số lẻ.
- Các phần tử ở 4 hình tam giác của 4 góc đều có giá trị bằng 0.
- Các phần tử còn lại đều có giá trị khác 0.

Ví dụ bên dưới là ma trận "hình thoi" cấp 5.

0	0	**8**	0	0
0	**3**	**2**	**5**	0
4	**1**	**3**	**4**	**9**
0	**5**	**6**	**7**	0
0	0	**2**	0	0

Cho tập tin văn bản đầu vào có tên là INPUT.TXT với cấu trúc như sau:

- Dòng đầu của tập tin lưu trữ 1 số nguyên dương N lẻ ($3 \le N < 20$), quy định cấp của ma trận vuông.
- Từ dòng thứ 2 trở đi, mỗi dòng lưu N số nguyên, là nội dung từng dòng của ma trận.

Yêu cầu:

1) *(2 điểm)* Viết hàm mở và đọc dữ liệu từ tập tin INPUT.TXT trên, lưu vào biến N và ma trận M cấp NxN.

   Khai báo hàm gợi ý:

   `void docfile(char *filename, int **M, int &N);`

2) *(2 điểm)* Viết hàm kiểm tra xem ma trận M có phải là ma trận "hình thoi" hay không. Nếu có, hàm trả về trị 1, ngược lại trả về 0.

   Khai báo hàm gợi ý: **`int mthinhthoi(int **M, int N);`**

3) *(2 điểm)* Giả sử M là ma trận "hình thoi", viết hàm tính tổng các phần tử nằm trên 4 cạnh của "hình thoi". Trong ví dụ trên, tổng các phần tử trên 4 cạnh bằng 43.

   Khai báo hàm gợi ý: **`int tongcanh(int **M, int N);`**

Tự chọn 1:

4) *(2 điểm)* Giả sử M là ma trận "hình thoi", hãy cài đặt một giải thuật theo phương pháp tham lam (Greedy) để tìm một đường đi từ **đỉnh trên (ô [0][N/2]** xuống **đỉnh dưới (ô [N-1][N/2]** của "hình thoi", sao cho tổng giá trị các phần tử nằm trên đường đi là **lớn nhất**. Nguyên tắc đi như sau:

- Ở mỗi bước, chỉ có thể đi xuống dòng liền kề với dòng hiện tại và chỉ được chọn 1 trong 3 vị trí gần vị trí hiện tại nhất. Tức là, từ vị trí hiện tại (i, j), ta chỉ có thể đi đến các vị trí (i+1, j-1), (i+1, j) hoặc (i+1, j+1).
- Không được đi vào các ô có giá trị bằng 0 (chỉ di chuyển nội trong phạm vi của "hình thoi")

Khai báo hàm gợi ý: `int thamlam(int **M, int N);`

5) (2 điểm) Hãy cài đặt hàm sử dụng phương pháp quay lui/vét cạn để tìm đường đi có tổng lớn nhất thực sự. Quy tắc đi giống câu 4).

Khai báo hàm gợi ý: `int vetcan(int **M, int N);`

VD: các ô tô đậm là những ô nằm trên đường đi. Tổng đường đi theo phương pháp tham lam bằng 25, theo phương pháp vét cạn bằng 27.

0	0	**8**	0	0
0	**3**	**2**	**4**	0
9	**1**	**5**	**4**	**3**
0	**5**	**4**	**6**	0
0	0	**2**	0	0

0	0	**8**	0	0
0	**3**	**2**	**4**	0
9	**1**	**5**	**4**	**3**
0	**5**	**4**	**6**	0
0	0	**2**	0	0

Tự chọn 2:

6) *(2.5 điểm)* Giả sử M là ma trận "hình thoi", một đường đi từ **đỉnh trên (ô [0][N/2]** xuống **đỉnh dưới (ô [N-1][N/2]** của "hình thoi" được xác định theo các nguyên tắc đi như sau:
   - Ở mỗi bước, chỉ có thể đi xuống dòng liền kề với dòng hiện tại và chỉ được chọn 1 trong 3 vị trí gần vị trí hiện tại nhất. Tức là, từ vị trí hiện tại (i, j), ta chỉ có thể đi đến các vị trí (i+1, j-1), (i+1, j) hoặc (i+1, j+1).
   - Không được đi vào các ô có giá trị bằng 0 (chỉ di chuyển nội trong phạm vi của "hình thoi")

   Hãy trình bày ý tưởng, sau đó cài đặt thuật toán để tìm đường đi có tổng giá trị các phần tử nằm trên đường đi là **lớn nhất (thực sự).**

   Khai báo hàm gợi ý: `int duongdimax(int **M, int N);`

7) (1.5 điểm) Phân tích và cho biết độ phức tạp của thuật toán đã cài đặt trong câu trên.

-HẾT-

```cpp
ï»¿// Há»  vÃ  tÃªn: LÃª Trá» ng DÅ©ng
// MSSV:21110157
// NgÃ y cáº-p nháº-t cuá»‘i:20/05/2022
// CÃ´ng dá»¥ng : bÃ i táº-p ktlt Ä‘á»  2015-2016
#include<iostream>
#include<iomanip>
using namespace std;
#define MAX 20

//Khai bÃ¡o biáº¿n toÃ n cá»¥c
int minpath[MAX];
int tongmax = 0;
int path[MAX];
int tong = 0;
//Khai bÃ¡o hÃ m
//1. HÃ m in ma tráº-n
void inmt(int M[][MAX], int n);
//2. HÃ m Ä‘á» c dá»¯ liá»‡u tá»« táº-p tin, lÆ°u vÃ o ma tráº-n
void docfile(const char filename[], int M[][MAX], int& n);
//3. HÃ m kiá»ƒm tra ma tráº-n cÃ³ pháº£i lÃ  hÃ¬nh thoi hay khÃ´ng
int ktmthinhthoi(int M[][MAX], int n);
//4. hÃ m tÃ¬m Ä‘Æ°á» ng Ä‘i cÃ³ tá»•ng lá»›n nháº¥t báº±ng tham lam
int thamlam(int M[][MAX], int n, int path[]);
//5.HÃ m tÃ¬m Ä‘Æ°á» ng cÃ³ tá»•ng lá»›n nháº¥t báº±ng vÃ©t cáº¡n
int vetcan(int M[][MAX], int n, int p[]);
//6. HÃ m quay lui Ä‘á»ƒ tÃ¬m Ä‘Æ°á» ng
void backtracking(int M[][MAX], int n, int i, int k);

int main()
{
    int M[MAX][MAX];
    int n;
    docfile("D:/INPUT.TXT", M, n);
    inmt(M, n);
    if (ktmthinhthoi(M, n))
    {
        cout << "Day la ma tran hinh thoi!" << endl;
        int path[50];
        int t = thamlam(M, n, path);
        cout << "Tong duong di theo pp tham lam la: " << t << endl;
        cout << "Duong di: ";
        for (int i = 0; i < n; i++)
```

```
37          cout << "Day la ma tran hinh thoi!" << endl;
38          int path[50];
39          int t = thamlam(M, n, path);
40          cout << "Tong duong di theo pp tham lam la: " << t << endl;
41          cout << "Duong di: ";
42          for (int i = 0; i < n; i++)
43              cout << "(" << i << "," << path[i] << ")" << "-->";
44          cout << "end" << endl;
45
46          int tong = vetcan(M, n, path);
47          cout << "Tong duong di lon nhat theo pp vet can la la: " << tong << endl;
48          cout << "Duong di lÃ : ";
49          for (int i = 0; i < n; i++)
50              cout << "(" << i << "," << path[i] << ")" << "-->";
51          cout << "end" << endl;
52      }
53      else
54          cout << "khong phai la ma tran hinh thoi." << endl;
55      return 0;
56  }
57  //1. HÃ m in ma tráº-n
58  void inmt(int M[][MAX], int n)
59  {
60      for (int i = 0; i < n; i++)
61      {
62          for (int j = 0; j < n; j++)
63          {
64              cout.width(4);
65              cout << M[i][j];
66          }
67          cout << endl;
68      }
69  }
70  //2. HÃ m Ä‘á» c dá»¯ liá»‡u tá»« tá»-p tin, lÆ°u vÃ o ma tráº-n
71  void docfile(const char filename[], int M[][MAX], int& n)
72  {
73      FILE* fp;
74      fopen_s(&fp, filename, "rt");
75      if (fp == NULL)
76      {
77          cout << "Khong mo duoc tap tin!\n";
78          return;
```

```
    {
        cout << "Khong mo duoc tap tin!\n";
        return;
    }
    // Ä‘á» c dá»¯ liá»‡u
    fscanf_s(fp, "%d", &n);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            fscanf_s(fp, "%d", &M[i][j]);
    // Ä‘Ã³ng táº-p tin láº¡i
    fclose(fp);

}
//3. HÃ m kiá»fm tra ma tráº-n cÃ³ pháº£i lÃ  hÃ¬nh thoi hay khÃ´ng
int ktmthinhthoi(int M[][MAX], int n)
{
    for (int i = 0; i <= n / 2; i++)
        for (int j = 0; j < n; j++)
        {
            if (j >= n / 2 - i && j <= n / 2 + i)
            {
                if ((M[i][j]) == 0)
                    return 0;
                if ((M[n - 1 - i][j]) == 0)
                    return 0;
            }
            else
            {
                if ((M[i][j]) != 0)
                    return 0;
                if ((M[n - 1 - i][j]) != 0)
                    return 0;
            }
        }
    return 1;
}
//4. hÃ m tÃ¬m Ä‘Æ°á» ng Ä‘i cÃ³ tá»•ng lá»›n nháº¥t báº±ng tham lam
int thamlam(int M[][MAX], int n, int path[])
{
    int tong = M[0][n / 2]; //Ä‘á»&nh trÃªn
    int k = n / 2;
    path[0] = k;
```

```
//4. hÃ m tÃ¬m Ä‘Æ°á» ng Ä‘i cÃ³ tá»•ng lá»›n nhá¹¥t bá¹±ng tham lam
int thamlam(int M[][MAX], int n, int path[])
{
    int tong = M[0][n / 2]; //Ä‘á»‹nh trÃªn
    int k = n / 2;
    path[0] = k;
    for (int i = 1; i < n; i++) //xÃ©t tá»«ng dÃ²ng
    {
        int max = 0, v = -1;        //Ã´ lá»›n nhá¹¥t trong 3 Ã´
        for (int j = -1; j <= 1; j++)
            if ((M[i][k+j]) > max)
            {
                max = M[i][k + j];
                v = k + j;
            }
        tong += max;    //lÆ°u lá¹¡i Ä‘Æ°á» ng Ä‘i
        path[i] = v;
        k = v;
    }
    return tong;

}
//5.HÃ m tÃ¬m Ä‘Æ°á» ng cÃ³ tá»•ng lá»›n nhá¹¥t
int vetcan(int M[][MAX], int n, int p[])
{
    tong = M[0][(n / 2)];               //Ä‘á»‹nh trÃªn
    for (int k = 0; k < n; k++)
    {
        //  tong += M[0][k];
        path[0] = n/2;
        backtracking(M, n, 1, k);
    }
    memcpy(p, minpath, n * sizeof(int));
    return tongmax;
}

//6. HÃ m quay lui Ä‘á»ƒ tÃ¬m Ä‘Æ°á» ng
void backtracking(int M[][MAX], int n, int i, int k)
{

    if (i >= n)  //xem tá»›i Ä‘Ã-ch hay chÆ°a
    {
```

```
    for (int k = 0; k < n; k++)
    {
        //  tong += M[0][k];
        path[0] = n/2;
        backtracking(M, n, 1, k);
    }
    memcpy(p, minpath, n * sizeof(int));
    return tongmax;
}

//6. HÃ m quay lui Ä‘á»ƒ tÃ¬m Ä‘Æ°á»ng
void backtracking(int M[][MAX], int n, int i, int k)
{

    if (i >= n)  //xem tá»›i Ä‘Ã-ch hay chÆ°a
    {
        if (tong > tongmax) {
            tongmax = tong;
            memcpy(minpath, path, n * sizeof(int));
        }
        return;
    }
    for (int j = -1; j <= 1; j++)
        if (k + j >= 0 && k + j < n && M[i][k + j] > 0)
        {
            tong += M[i][k + j];
            path[i] = k + j;
            backtracking(M, n, i + 1, k + j);       //Gá»i Ä‘á»‡ quy Ä‘á»ƒ Ä‘i tiáº¿p tá»« Ã´ Ä‘Ã³
            tong -= M[i][k + j];                    //tráº£ láº¡i tá»•ng trÆ°á»›c khi thá» Ä‘i
        }
}
```

CHUỖI LŨY THỪA CỦA MỘT SỐ HÀM CƠ BẢN

Tên gọi	Chuỗi	Khoảng hội tụ
Chuỗi hàm mũ	$e^u = 1 + u + \frac{u^2}{2!} + \frac{u^3}{3!} + \ldots + \frac{u^k}{k!} + \ldots = \sum_{k=0}^{\infty} \frac{u^k}{k!}$	$(-\infty, \infty)$
Chuỗi Cosin	$\cos u = 1 - \frac{u^2}{2!} + \frac{u^4}{4!} - \ldots + \frac{(-1)^k u^{2k}}{(2k)!} + \ldots = \sum_{k=0}^{\infty} \frac{(-1)^k u^{2k}}{(2k)!}$	$(-\infty, \infty)$
Chuỗi Sin	$\sin u = u - \frac{u^3}{3!} + \frac{u^5}{5!} - \ldots + \frac{(-1)^k u^{2k+1}}{(2k+1)!} + \ldots = \sum_{k=0}^{\infty} \frac{(-1)^k u^{2k+1}}{(2k+1)!}$	$(-\infty, \infty)$
Chuỗi nghịch đảo	$\frac{1}{1-u} = 1 + u + u^2 + u^3 + \ldots + u^k + \ldots = \sum_{k=0}^{\infty} u^k$	$(-1,1)$
Chuỗi Logarit	$\ln(1+u) = u - \frac{u^2}{2} + \frac{u^3}{3} - \ldots + \frac{(-1)^{k-1} u^k}{k} + \ldots = \sum_{k=1}^{\infty} \frac{(-1)^{k-1} u^k}{k}$	$(-1,1]$
Chuỗi $\tan^{-1}$	$\tan^{-1} u = u - \frac{u^3}{3} + \frac{u^5}{5} - \ldots + \frac{(-1)^k u^{2k+1}}{(2k+1)} + \ldots = \sum_{k=0}^{\infty} \frac{(-1)^k u^{2k+1}}{(2k+1)}$	$[-1,1]$
Chuỗi $\sin^{-1}$	$\sin^{-1} u = u + \frac{u^3}{2.3} + \frac{1.3 u^5}{2.4.5} + \frac{1.3.5 u^7}{2.4.6.7} + \ldots + \frac{1.3.5 \ldots (2k-3) u^{2k-1}}{2.4.6 \ldots (2k-2)(2k-1)} + \ldots$	$[-1,1]$
Chuỗi nhị thức	$(1+u)^p = 1 + pu + \frac{p(p-1)}{2!} u^2 + \frac{p(p-1)(p-2)}{3!} u^3 + \ldots$	$[-1,1]$