# CÁCH ĐÁNH GIÁ ĐIỂM THỰC HÀNH

## HỌC PHẦN: IT3150 – Project 1- 2023.1

## I. Quy định, yêu cầu:

- Tài liệu và nội dung thực hành chấm điểm trên hệ thống: https://lab.soict.hust.edu.vn/
- Bài tập trên lớp chấm điểm tự động (các bài không chấm trên hệ thống làm vào máy tính ➔ làm báo cáo thực hành – Theo mẫu).
- Hạn nộp báo cáo trên Teams (Bài tập trên lớp + Bài tập về nhà): 1 tuần.

## II. Đánh giá điểm thực hành

1. Chuyên cần (đúng giờ, nghiêm túc trong giờ học) - Điểm danh trên Teams: 10%
2. Báo cáo thực hành (bài tập trên lớp + Về nhà) theo mẫu nộp trên Teams: 40%
3. Trắc nghiệm – Form trên Teams: 10%
4. Kiểm tra thực hành: 40%. (Tiết 2,3 buổi thực hành thứ 5).

**Điểm thưởng: 5% ➔ 10% (Cho Mục 1,2 điểm TB từ 9-10).**

Tham gia thực hành đúng giờ đầy đủ theo thời khóa biểu (nếu có lý do không đi thực hành đúng kíp được thì gửi mail xin phép thực hành bù trước 1 ngày qua mail hoalt@soict.hust.edu.vn, Tiêu đề: đăng ký học bù – IT3040 – MaLopTH.
Các kíp có thể bù:

| TT | Thời gian, địa điểm, Tuần học | Mã nhóm | Mã lớp |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

**Nếu nghỉ không có lý do 3 buổi, không thực hành bù thì điểm chuyên cần, báo cáo và BTVN coi như 0 điểm thực hành.**

# Contents

# Bài 1: Problem: Dãy fibonacci

---

Cho dãy fibonacci a[0], a[1], a[2], ... trong đó: a[0] = 0, a[1] = 1, a[n] = a[n-1] + a[n-2], với mọi n >= 2

Cho số nguyên dương n, tính a[n-1].

**Đầu vào**

Dòng 1: chứa số nguyên dương n (2 <= n <= 21)

**đầu ra**

Viết một [n-1]

**Ví dụ**

**Đầu vào**

9

**đầu ra**

21

Source code

```cpp
//Mai Minh Hoàng
//20215381
#include <bits/stdc++.h>
using namespace std;
Long Fibonacci_81(int n_81){ // Hàm Fibonacci_81 tính số Fibonacci thứ n_81
    if(n_81==1 || n_81==0){ // Trường hợp cơ sở
        return n_81;
    }
    return Fibonacci_81(n_81-1)+Fibonacci_81(n_81-2); // Tính số Fibonacci thứ n_81
}
int main()  // Hàm main
{
    int n_81; // Khai báo biến n_81
    cin>>n_81; // Nhập giá trị từ bàn phím
    cout<<Fibonacci_81(n_81-1); // In ra số Fibonacci thứ (n_81 - 1)
    return 0;
}
```

# Test:

Input

16

Correct output

610

User output

610

Input

30

Correct output

514229

User output

514229

Input

21

Correct output

6765

User output

6765

## Bài 2: Vấn đề: Tính C_k_n

---

Cho hai số nguyên dương k và n. Tính C(k,n) là số cách để chọn k đối tượng từ một tập hợp n đối tượng nhất định.

### Đầu vào

- Dòng 1: hai số nguyên dương k và n (1 <= k,n <= 999)

### đầu ra

Viết giá trị te C(k,n) modulo 109+7.

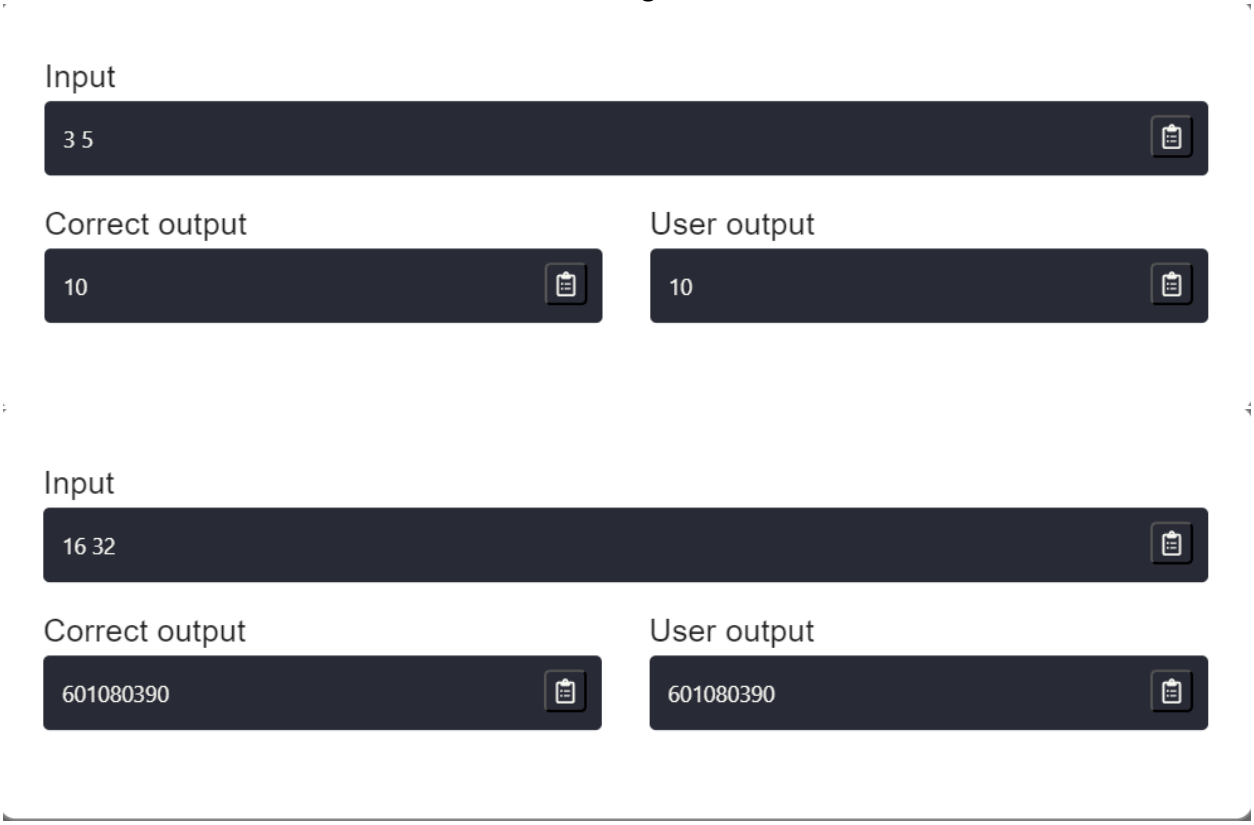
Source code

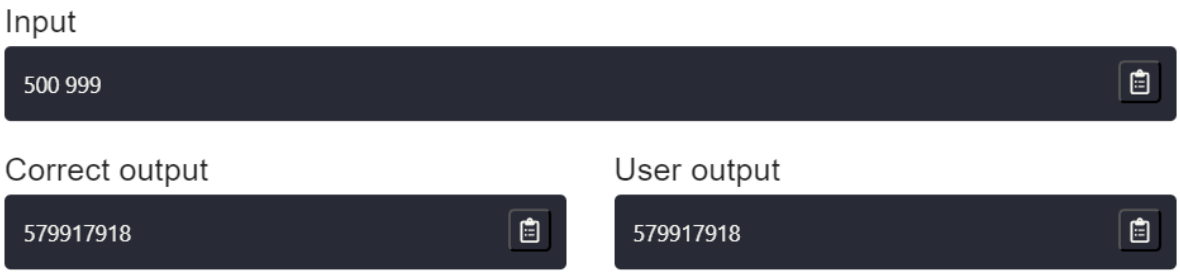
C++ 17

```cpp
//Mai Minh Hoàng
//20215381
#include <bits/stdc++.h>
using namespace std;
#define N_81 1000
long long mang_81[N_81][N_81]; // Khai báo mảng hai chiều mang_81
long long toHop_81(long long n, long long k) { // Hàm toHop_81 tính tổ hợp chập k của n
    if (n == 1 || k==0 ||k==n) { // Trường hợp cơ sở
        return 1;
    }
    if (mang_81[n][k] != 0) { // Trường hợp đã tính toán trước đó
        return mang_81[n][k];
    }
    mang_81[n][k] = (toHop_81(n - 1, k - 1) + toHop_81(n - 1, k))%1000000007; // Tính toán tổ hợp chập k của
    return mang_81[n][k]; // Trả về kết quả
}
int main() {
    long long n, k;
    cin >> k >> n;  // Nhập giá trị từ bàn phím
    cout << toHop_81(n, k); // In ra tổ hợp chập k của n
    return 0;
}

```

### Test:




## Bài 3: Vấn đề: Tạo chuỗi nhị phân

---

Cho một số nguyên n, hãy viết chương trình tạo ra tất cả các chuỗi nhị phân có độ dài n theo thứ tự từ điển.

### Đầu vào

- Dòng 1: chứa số nguyên n (1 <= n <= 20)

### đầu ra

# Viết các chuỗi nhị phân theo thứ tự từ điển, mỗi chuỗi trên một dòng

Source code C++

```cpp
//Mai Minh Hoàng
//20215381
#include <bits/stdc++.h>
using namespace std;
#define N_81 20 // Định nghĩa hằng số N_81
int mang_81[N_81]; // Khai báo mảng một chiều mang_81
void printt_81(int k_81,int n_81) { // Hàm printt_81 in ra tất cả các chuỗi nhị phân độ dài n_81
    if (k_81 == n_81) { // Trường hợp cơ sở
        for (int i = 0; i < n_81; i++) { // Duyệt qua từng phần tử của mảng
            cout << mang_81[i]; // In ra giá trị của phần tử thứ i
        }
        cout << endl;
        return; // Kết thúc hàm
    }
    for (int i = 0; i <= 1; i ++) { // Duyệt qua từng giá trị có thể của phần tử thứ k_81
        mang_81[k_81] = i; // Gán giá trị cho phần tử thứ k_81
        printt_81(k_81+1, n_81); // Gọi đệ quy hàm printt_81 với k_81 tăng lên 1 đơn vị
    }
}
int main() {
    int n_81; // Khai báo biến n_81
    cin >> n_81; // Nhập giá trị từ bàn phím
    printt_81(0, n_81); // Gọi hàm printt_81 để in ra tất cả các chuỗi nhị phân độ dài n_81
    return 0;
}

```

## Test:

Input

```
10
```

Correct output

```
0000000000
0000000001
0000000010
0000000011
0000000100
0000000101
0000000110
0000000111
0000001000
0000001001
0000001010
0000001011
0000001100
0000001101
```

User output

```
0000000000
0000000001
0000000010
0000000011
0000000100
0000000101
0000000110
0000000111
0000001000
0000001001
0000001010
0000001011
0000001100
0000001101
```


## Bài 4: Vấn đề: Tạo chuỗi nhị phân không có số 11 liên tiếp

---

Cho một số nguyên n, hãy viết chương trình tạo ra tất cả các chuỗi nhị phân không có số 11 liên tiếp theo thứ tự từ điển.

## Đầu vào

- Dòng 1: chứa số nguyên n (1 <= n <= 20)

## đầu ra

Viết các chuỗi nhị phân theo thứ tự từ điển, mỗi chuỗi trên một dòng

## Ví dụ

## Đầu vào

3

## đầu ra

000

001

010

100

101

## Source code

```cpp
//Mai Minh Hoàng
//20215381
#include <bits/stdc++.h>
using namespace std;

#define N_81 20 // Định nghĩa hằng số N_81

int mang_81[N_81]; // Khai báo mảng mang_81 với kích thước N_81
int maxs_81 = -1; // Khai báo biến maxs_81 và gán giá trị ban đầu là -1

// Hàm printt_81 để in ra các số nhị phân có độ dài n_81
void printt_81(int k_81, int n_81) {
    if (k_81 == n_81) { // Nếu đã đủ n_81 chữ số
        int s_81 = 0;
        // Tính giá trị của số nhị phân hiện tại
        for (int i_81 = 0; i_81 < n_81; i_81++) {
            s_81 += mang_81[i_81] * pow(2, n_81 - i_81 - 1);
        }
        if (s_81 > maxs_81) { // Nếu giá trị mới lớn hơn giá trị lớn nhất đã thấy
            for (int i_81 = 0; i_81 < n_81; i_81++) {
                cout << mang_81[i_81]; // In ra số nhị phân
            }
            cout << endl;
            maxs_81 = s_81; // Cập nhật giá trị lớn nhất
        }
        return;
    }
    for (int i_81 = 0; i_81 <= 1; i_81++) { // Duyệt qua cả hai giá trị 0 và 1
        if (mang_81[k_81 - 1] != 1 || i_81 != 1) {
            mang_81[k_81] = i_81;
        }
        else {
            mang_81[k_81] = 0;
        }
        printt_81(k_81 + 1, n_81); // Gọi đệ quy để xây dựng số nhị phân tiếp theo
    }
}

int main() {
    int n_81;
    cin >> n_81; // Nhập độ dài của số nhị phân
    printt_81(0, n_81); // Gọi hàm để in ra tất cả các số nhị phân có độ dài n_81
    return 0;
}
```

# Test:

Input

```
5
```

Correct output

```
00000
00001
00010
00100
00101
01000
01001
01010
10000
10001
10010
10100
10101
```

User output

```
00000
00001
00010
00100
00101
01000
01001
01010
10000
10001
10010
10100
10101
```

Input

```
3
```

Correct output

```
000
001
010
100
101
```

User output

```
000
001
010
100
101
```


## Bài 5: Vấn đề: Tạo hoán vị

---

Cho một số nguyên n, viết chương trình tạo tất cả các hoán vị 1, 2, ..., n theo thứ tự từ vựng (các phần tử của hoán vị được phân tách bằng ký tự SPACE).

**Ví dụ**

**Đầu vào**

3

**đầu ra**

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

Source code

```cpp
//Mai Minh Hoàng
//20215381
#include <bits/stdc++.h>
using namespace std;
#define N_81 20 // Định nghĩa hằng số N_81
int mang_81[N_81]; // Khai báo mảng một chiều mang_81
int ktra_81[N_81]; // Khai báo mảng một chiều ktra_81
void printt_81(int k_81, int n_81) { // Hàm printt_81 in ra tất cả các hoán vị của n phần tử
    if (k_81 == n_81) { // Trường hợp cơ sở
        for (int i = 0; i < n_81; i++) { // Duyệt qua từng phần tử của mảng
            cout << mang_81[i]<<" "; // In ra giá trị của phần tử thứ i
        }
        cout << endl; // Xuống dòng
        return; // Kết thúc hàm
    }
    for (int i = 1; i <= n_81; i++) { // Duyệt qua từng giá trị có thể của phần tử thứ k_81
        if (ktra_81[i] == 0) { // Nếu giá trị chưa được sử dụng
            mang_81[k_81] = i; // Gán giá trị cho phần tử thứ k_81
            ktra_81[i] = 1; // Đánh dấu giá trị đã được sử dụng
            printt_81(k_81 + 1,  n_81); // Gọi đệ quy hàm printt với k tăng lên 1 đơn vị
            ktra_81[i] = 0; // Bỏ đánh dấu giá trị đã được sử dụng
        }
        else {
            continue; // Bỏ qua lần lặp hiện tại và tiếp tục với lần lặp tiếp theo
        }
    }
}
int main() {
    int n_81;
    cin >>  n_81; // Nhập giá trị từ bàn phím
    printt_81(0,  n_81); // Gọi hàm printt để in ra tất cả các hoán vị của n phần tử
    return 0;
}
```

# Test:

Input

```
6
```

Correct output

```
1 2 3 4 5 6
1 2 3 4 6 5
1 2 3 5 4 6
1 2 3 5 6 4
1 2 3 6 4 5
1 2 3 6 5 4
1 2 4 3 5 6
1 2 4 3 6 5
1 2 4 5 3 6
1 2 4 5 6 3
1 2 4 6 3 5
1 2 4 6 5 3
1 2 5 3 4 6
```

User output

```
1 2 3 4 5 6
1 2 3 4 6 5
1 2 3 5 4 6
1 2 3 5 6 4
1 2 3 6 4 5
1 2 3 6 5 4
1 2 4 3 5 6
1 2 4 3 6 5
1 2 4 5 3 6
1 2 4 5 6 3
1 2 4 6 3 5
1 2 4 6 5 3
1 2 5 3 4 6
```

Input

```
3
```

Correct output

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

User output

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

Input

```
8
```

Correct output

```
12345678
12345687
12345768
12345786
12345867
12345876
12346578
12346587
12346758
12346785
12346857
12346875
12347568
```

User output

```
12345678
12345687
12345768
12345786
12345867
12345876
12346578
12346587
12346758
12346785
12346857
12346875
12347568
```

## Bài 6: Bài toán: Đếm số đáp án sudoku

---

Viết chương trình tính số đáp án sudoku (điền các phần tử 0 của bảng sudoku từng phần cho trước)

Điền các số từ bảng 1, 2, 3,.., 9 đến 9 x 9 sao cho:

- Số mỗi hàng khác nhau
- Số mỗi cột khác nhau
- Các số trên mỗi ô vuông con 3 x 3 khác nhau

### Đầu vào

- Mỗi dòng *i* ( *i* = 1, 2, ..., 9) chứa các phần tử của *i*quần quèhàng của bảng Sudoku: các phần tử là các số từ 0 đến 9 (giá trị 0 nghĩa là ô trống của bảng)

### đầu ra

- Viết số phương án tìm được

Source code

C++ 17

```cpp
//Mai Minh Hoàng
//20215381
#include <bits/stdc++.h>
using namespace std;
int mang_81[9][9]; // Khai báo mảng hai chiều mang_81
int dem_81=0; // Khai báo và khởi tạo biến đếm dem_81
bool checkrow_81(int row_81,int i_81) { // Hàm checkrow_81 kiểm tra hàng row_81 có chứa số i_81 hay không
    for (int k = 0; k < 9; k++) { // Duyệt qua từng phần tử của hàng
        if (mang_81[row_81][k] == i_81) { // Nếu phần tử thứ k của hàng row_81 bằng i_81
            return false; // Trả về false
        }
    }
    return true; // Trả về true
}
bool checkcol_81(int col_81,int i_81){ // Hàm checkcol_81 kiểm tra cột col_81 có chứa số i_81 hay không
    for (int k = 0; k < 9; k++) { // Duyệt qua từng phần tử của cột
        if (mang_81[k][col_81] == i_81) { // Nếu phần tử thứ k của cột col_81 bằng i_81
            return false; // Trả về false
        }
    }
    return true; // Trả về true
}
bool check3x3_81(int col, int row, int i) { // Hàm check3x3 kiểm tra ô vuông 3x3 chứa ô (row, col) có chứa
    int startrow = 3 * (row / 3); // Tính chỉ số hàng đầu tiên của ô vuông 3x3
```

```cpp
}
bool check3x3_81(int col, int row, int i) { // Hàm check3x3 kiểm tra ô vuông 3x3 chứa ô (row, col) có chứ
    int startrow = 3 * (row / 3); // Tính chỉ số hàng đầu tiên của ô vuông 3x3
    int startcol = 3 * (col / 3); // Tính chỉ số cột đầu tiên của ô vuông 3x3
    for (int k = startrow; k < startrow + 3; k++) { // Duyệt qua từng hàng của ô vuông 3x3
        for (int h = startcol; h < startcol + 3; h++) { // Duyệt qua từng cột của ô vuông 3x3
            if (mang_81[k][h] == i) { // Nếu phần tử tại vị trí (k, h) bằng i
                return false; // Trả về false
            }
        }
    }
    return true; // Trả về true
}
void total_81(int r,int c) {//Hàm đếm xem có bao nhiêu kết quả thỏa mãn
    if (r == 8 && c == 8) {//nếu là phần tử cuối của bảng
        for (int i = 1; i <= 9; i++) {
            if (check3x3_81(c, r, i) && checkcol_81(c, i) && checkrow_81(r, i)) {
                dem_81++;
            }
        }
        return;
    }
    if (mang_81[r][c] == 0) {
        for (int i = 1; i <= 9; i++) {
            if (check3x3_81(c,r, i) && checkcol_81(c, i) && checkrow_81(r,i)) {
```

```
    if (mang_81[r][c] == 0) {
        for (int i = 1; i <= 9; i++) {
            if (check3x3_81(c,r, i) && checkcol_81(c, i) && checkrow_81(r,i)) {
                mang_81[r][c] = i;

                if (c == 8) {
                    total_81(r + 1, 0);
                }
                else {

                    total_81(r, c + 1);
                }
                mang_81[r][c] = 0;
            }

        }

    }
    else {
        if (c == 8) {
            total_81(r + 1, 0);
        }
        else {
```

```
    }
    else {
        if (c == 8) {
            total_81(r + 1, 0);
        }
        else {

            total_81(r, c + 1);
        }
    }
}
int main() {
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            cin >> mang_81[i][j];//nhập ma trận ban đầu
        }
    }
    total_81(0, 0);//bắt đầu đếm xem có bao nhiêu trường hợp
    cout << dem_81;//in ra kết quả
    return 0;
}
```

# Test:

Input

```
003400089
006789023
080023456
004065097
060090014
007204365
030602078
000000000
000000000
```

Correct output

```
64
```

User output

```
64
```

Input

```
103400089
406789023
780023456
204065097
360090014
807004300
530602078
000000000
000000000
```

Correct output

```
8
```

User output

```
8
```

Input

```
103400089
406789023
780023000
204065097
360090014
807200300
530602078
000000000
000000000
```

Correct output

```
24
```

User output

```
24
```

## Bài 7: Phương trình số nguyên tuyến tính - hệ số 1

---

Cho hai số nguyên n và M. Viết chương trình sinh ra tất cả các tập hợp có thứ tự (X1, X2, ..., Xn) sao cho: X1 + X2 + . . . + Xn = M

**Đầu vào**

- Dòng 1: chứa 2 số nguyên n và M (2 <= n <= 10, 1 <= M <= 20)

**đầu ra**

- Viết vào mỗi dòng X1, X2, ..., Xn cách nhau bằng ký tự SPACE

Source code

```cpp
//Mai Minh Hoàng
//20215381
#include <bits/stdc++.h>
using namespace std;
#define N_81 20 // Định nghĩa hằng số N_81
int mang_81[N_81]; // Khai báo mảng một chiều mang_81
int sum_81 = 0; // Khai báo và khởi tạo biến tổng sum_81
int targetsum_81; // Khai báo biến tổng mục tiêu targetsum_81
void printt_81(int k_81, int n_81) { // Hàm printt_81 in ra tất cả các dãy số có tổng bằng targetsum_81
    if (k_81 == n_81) { // nếu số phần tử trong mảng đã đầy
        if (targetsum_81 == sum_81) { // Nếu tổng hiện tại bằng tổng mục tiêu

            for (int i = 0; i < n_81; i++) { // Duyệt qua từng phần tử của mảng
                cout << mang_81[i]<<" "; // In ra giá trị của phần tử thứ i
            }
            cout << endl; // Xuống dòng
        }
        return; // Kết thúc hàm
    }
    for (int i = 1; i <= targetsum_81-sum_81; i++) { // Duyệt qua từng giá trị có thể của phần tử thứ k_81
        mang_81[k_81] = i; // Gán giá trị cho phần tử thứ k_81
        sum_81 += i; // Cập nhật tổng hiện tại
        printt_81(k_81 + 1, n_81); // Gọi đệ quy hàm printt với k tăng lên 1 đơn vị
        sum_81 -= i; // Trừ đi giá trị vừa thêm vào từ tổng hiện tại
        mang_81[k_81] = 0; // Đặt lại giá trị của phần tử thứ k về 0
    }
}
int main() {
    int n_81;
    cin >> n_81>>targetsum_81; // Nhập giá trị từ bàn phím
    printt_81(0, n_81);
    return 0;
}
```

# Test:

Input

```
3 5
```

Correct output

```
1 1 3
1 2 2
1 3 1
2 1 2
2 2 1
3 1 1
```

User output

```
1 1 3
1 2 2
1 3 1
2 1 2
2 2 1
3 1 1
```

Input

```
5 10
```

Correct output

```
1 1 1 1 6
1 1 1 2 5
1 1 1 3 4
1 1 1 4 3
1 1 1 5 2
1 1 1 6 1
1 1 2 1 5
1 1 2 2 4
1 1 2 3 3
1 1 2 4 2
1 1 2 5 1
1 1 3 1 4
1 1 3 2 3
1 1 3 3 2
```

User output

```
1 1 1 1 6
1 1 1 2 5
1 1 1 3 4
1 1 1 4 3
1 1 1 5 2
1 1 1 6 1
1 1 2 1 5
1 1 2 2 4
1 1 2 3 3
1 1 2 4 2
1 1 2 5 1
1 1 3 1 4
1 1 3 2 3
1 1 3 3 2
```

Input

```
6 11
```

Correct output

```
1 1 1 1 1 6
1 1 1 1 2 5
1 1 1 1 3 4
1 1 1 1 4 3
1 1 1 1 5 2
1 1 1 1 6 1
1 1 1 2 1 5
1 1 1 2 2 4
1 1 1 2 3 3
1 1 1 2 4 2
1 1 1 2 5 1
1 1 1 3 1 4
1 1 1 3 2 3
1 1 1 3 3 2
```

User output

```
1 1 1 1 1 6
1 1 1 1 2 5
1 1 1 1 3 4
1 1 1 1 4 3
1 1 1 1 5 2
1 1 1 1 6 1
1 1 1 2 1 5
1 1 1 2 2 4
1 1 1 2 3 3
1 1 1 2 4 2
1 1 1 2 5 1
1 1 1 3 1 4
1 1 1 3 2 3
1 1 1 3 3 2
```