智能算法挑战复赛初中组
（总共 4 道题）

1. 输出多进制数

【题目描述】
输入一个小于 20 的正整数 n，要求按从小到大的顺序输出所有的 n 位 m 进制数，
每个数占一行。

【输入格式】
输入一个小于 20 的正整数 n，和一个小于 10 的正整数 m。

【输出格式】
按从小到大的顺序输出所有的 n 位 m 进制数，每个数占一行。

【样例输入】（测试数据不包含本样例）
3
2

【样例输出】
000
001
010
011
100
101
110
111

答案：

```
#include <iostream>
#include <cstring>
#include <cmath>

using namespace std;

void ans(int n, int m) {
    string number(n, '0');
    for (int i = 0; i < pow(m, n); ++i) {
       int base = 1;
        for (int j = n - 1; j >= 0; --j) {
            number[j] = '0' + (i / base) % m;
            base *= m;
        }
        cout << number << endl;
    }
}

int main() {
```

```
    int n, m;
    cin >> n >> m;

    ans(n, m);
    return 0;
}
```

2. 自动驾驶汽车的路径规划问题

【题目描述】

一个自动驾驶的汽车，其只能按照调度系统到指定的景点停车场进行充电，每个景点中间的路上，都会有一些游客上车前往下一个景点。调度系统会告诉自动驾驶 汽车，到哪个景点的停车场上充电，但具体走那条路到该停车场充电没有限制，但 是自动驾驶汽车在每次行进过程中，不能重复到达同一个景点。请设计算法，给出 从景点 a 到景点 b 进行充电，再返还景点 a，这个自动驾驶汽车如何规划行进线路（线路上不能到达同一个景点多次），使得沿途接上的游客数量是最多的，并输出最多的上车的游客数量。

【输入格式】

第一行有两个数：n 和 m，其中 n 表示景点的数量，m 是景点间的单行道数量。

第二行到第 m+1 行，有三个整数：第一个数是起始景点的编号，第二个数是该路径终点景点的编号，第三个数是从起点到终点，需要乘车的人数。

第 m+2 行，有两个数字，第一个数字是自动驾驶汽车的出发景点，第二数字是中间进行充电的景点编号。

【输出格式】

一个数，表示该自动驾驶汽车往返一次，可以接送最多多少人。

【样例输入】（测试数据不包含本样例）

```
3  5
1  2  4
2  1  6
1  3  11
3  1  3
2  3  2
1  2
```

【样例输出】

```
10
```

答案：

```
#include <iostream>
#include <cmath>

using namespace std;

const int N = 1e3 + 9;
const int M = 1e5 + 9;

int n, m, start, to, ans;
int mp[N][N];
int vis[M];
int a[M];

void dfs(int now, int sum, int tot) {
    a[tot] = now;
```

```
    if(now == start) {
        int flag;
        flag = 0;
        for(int i = 1; i <= tot; i++) {
            if(a[i] == to) {
                flag = 1;
                break;
            }
        }
        if(!flag) {
            return;
        }
        if(sum > ans) {
            ans = sum;
        }
        return;
    }

    for(int i = 1; i <= n; i++) {
        if(mp[now][i] != 0 && !vis[i]) {
            vis[i] = 1;
            dfs(i, sum + mp[now][i], tot+1);
            vis[i] = 0;
        }
    }
}

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

    int u, v;
    for(int i = 1; i <= m; i++){
        cin >> u >> v;
        cin >> mp[u][v];
    }
    cin >> start >> to;


    for(int i = 1; i <= n; i++) {
        if(mp[start][i] != 0 && !vis[i]) {
            vis[i] = 1;
            a[1] = i;
            dfs(i, mp[start][i], 2);
            vis[i] = 0;
```

```
            }
        }

        cout << ans << endl;

        return 0;
}
```

3. 删除 k 位数字，得到最小的数

【题目描述】

输入一个数字串 N，长度不超过 250 位，去掉其中任意 k 个数字后剩下的数字按 原左右次序将组成一个新的整数，要求组成新的整数最小。

【输入格式】

输入两行正整数。第一行输入一个高精度的正整数 n。第二行输入一个正整数 k，表示需要删除的数字个数。

【输出格式】

输出一个整数，最后剩下的最小数。

【样例输入】（测试数据不包含本样例）

175438

4

【样例输出】

13

答案：

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
string fun(string num, int k) {
    string result;
    for (char c : num) {
        while (!result.empty() && k > 0 && result.back() > c) {
            result.pop_back();
            k--;
        }
        if (c != '0' || !result.empty()) {
            result.push_back(c);
        }
    }
    while (k > 0 && !result.empty()) {
        result.pop_back();
        k--;
    }
    return result;
}
int main() {
    string num;
    int k;
    cin >> num >> k;
    cout << fun(num, k) << endl;
    return 0;
}
```

4. 在 AI 下棋程序中，计算猫抓老鼠游戏的概率

【题目描述】

有这样一个游戏：在一个 n*n 的格子棋盘里，n 是奇数；有两种棋子，一个是只能横向移动的棋子猫，一个是可以上下左右移动的棋子老鼠。假设老鼠在棋盘的正 中央，第一步老鼠将进行上下左右的随机移动。棋子猫在从棋盘的中间行的最左边 向棋盘的最右边移动，棋子猫每次移动只能是从左到右移动一步，第一步是猫位于棋盘的中间行的最左边格子。请问：在猫移动到棋盘外面前，会有多大概率抓到老鼠？

【输入格式】

输入一个大于 1 的奇数 n，表示棋盘的大小。

【输出格式】

棋子猫抓到棋子老鼠的概率。（小数四舍五入保留 4 位有效数字）

【样例输入】（测试数据不包含本样例）

3

【样例输出】

0.6667

答案：

```
#include <iostream>
#include <cmath>

using namespace std;

const int N = 1e3 + 9;

int n, zhua;
double ans;

int dx[] = {0, 0, 1, -1};
int dy[] = {1, -1, 0, 0};
int catx;

void dfs(int caty, int x, int y, double gailu) {
    if(caty == y && catx == x) {
        ans += gailu;
        return;
    }
    if(caty == n+1) {
        return;
    }
    int nx, ny;

    for(int i = 0; i <= 3; i++) {
```

```
            nx = x + dx[i];
            ny = y + dy[i];
            if(nx <= 0 || nx > n || ny <= 0 || ny > n) {
                continue;
            }

            if((x == 1 && y != 1 && y != n) || (x == n && y != 1 && y != n) || (y == 1 &&
x != 1 && x != n) || (y == n && x != 1 && x != n)) {
                dfs(caty+1, nx, ny, gailu/3.0);
            }
            else if(x != 1 && x != n && y != 1 && y != n) {
                dfs(caty+1, nx, ny, gailu/4.0);
            }
            else {
                dfs(caty+1, nx, ny, gailu/2.0);
            }

        }
}

int main() {
    cin >> n;
    catx = (n+1)/2;

    dfs(0, (n+1)/2, (n+1)/2, 1);

    printf("%.4f", ans);

}
```