前言

本文档为2021秋季学年清华大学计算机系程序设计基础题目合集，任课老师为徐明星。

如无特殊说明所有题目时间限制为1000ms，内部限制为256MB。

文档中所给出的源代码并不是全部正确（有错误的在开头已标出），且许多算法仍可优化。

Week_12

P1687.n皇后问题——有坑版

Description:

Question:

> 给定一个n*n的棋盘，棋盘中有一些位置有坑，不能放皇后。现在要向棋盘中放入皇后，使得任意两个皇后都不在同一行、同一列或同一条对角线上。问最多能放多少个皇后?

Input:

> 输入的第一行为一个整数n，表示棋盘的大小。
> 接下来n行，每行n个0或1的整数，如果一个整数为1，表示对应的位置可以放皇后，如果一个整数为0，表示对应的位置不可以放皇后。

Output:

> 输出一个整数，表示最多能放多少个皇后。

Sample input:

```
4
1 0 0 0
0 1 0 0
0 0 0 1
1 0 0 1
```

Sample output:

```
3
```

Note:

> *data size:*
>
> 对于20%数据，n=4；
>
> 对于90%数据，n<=8；
> 对于100%数据，n<=13。
>
> 本题时间限制1s，内存限制256MB。

src:

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

int maxPlace = 0;
bool _try(int col, int n, int num, bool **place, bool *rowUnsafe, bool *LUnsafe, bool *RUnsafe)
{
    if (col == n)
    {
        num > maxPlace ? maxPlace = num : maxPlace = maxPlace;
        return true;
    }
    for (int row = 0; row < n + 1; row++)
    {
        if (row < n)
        {
            if (!place[row][col] || rowUnsafe[row] || LUnsafe[row + col] || RUnsafe[row - col + n - 1])
                continue;
            num++;
            rowUnsafe[row] = true;
            LUnsafe[row + col] = true;
            RUnsafe[row - col + n - 1] = true;
            _try(col + 1, n, num, place, rowUnsafe, LUnsafe, RUnsafe);
            num--;
            rowUnsafe[row] = false;
            LUnsafe[row + col] = false;
            RUnsafe[row - col + n - 1] = false;
        }
        else
            _try(col + 1, n, num, place, rowUnsafe, LUnsafe, RUnsafe);
    }
    return true;
}

int main()
{
    int n = 0;
    cin >> n;
    bool **place = new bool *[n]();
    for (int i = 0; i < n; i++)
        place[i] = new bool[n]();
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
        {
            int tmp = 0;
            cin >> tmp;
            if (tmp)
                place[i][j] = true;
        }
    bool *rowUnsafe = new bool[n]();
    bool *LUnsafe = new bool[2 * n - 1]();
    bool *RUnsafe = new bool[2 * n - 1]();
    _try(0, n, 0, place, rowUnsafe, LUnsafe, RUnsafe);
    cout << maxPlace << endl;
```

```
54        return 0;
55    }
```

P2341.下楼问题

Description:

Question:

> 从楼上走到楼下共有h个台阶，每一步有3种走法：走一个台阶；走两个台阶；走三个台阶。规定只能往下走，不能往上走。
>
> 调皮的小明在n个台阶上撒了水，为了防止滑倒，不能踏上这n个台阶，问从楼上到楼下可走出多少种方案?

Input:

> 第一行两个数字h, n，分别表示总台阶数和不能走的台阶数（1 <= n < h <= 15）
>
> 接下来n行，每行一个整数a_i，表示第a_i级台阶被撒了水不能走

Output:

> 一个整数，表示从楼上到楼下的方案数。

Sample input:

```
1    5 1
2    3
```

Sample output:

```
1    5
```

Note:

> 第0级和第h级台阶保证不会有水

---

src:

```
1    #include <iostream>
2    using namespace std;
3
4    void step(int cur, int h, bool *unWalk, int &total)
5    {
6        if (cur == h)
7        {
8            total++;
9            return;
10       }
11       for (int i = 0; i < 3; i++)
12       {
13           if (unWalk[cur + i] || cur + i + 1 > h)
```

```
            continue;
        cur += i + 1;
        step(cur, h, unWalk, total);
        cur -= i + 1;
    }
}

int main()
{
    int h = 0, n = 0;
    cin >> h >> n;
    bool *unWalk = new bool[h]();
    for (int i = 0; i < n; i++)
    {
        int tmp = 0;
        cin >> tmp;
        unWalk[tmp - 1] = true;
    }
    int total = 0;
    step(0, h, unWalk, total);
    cout << total << endl;
    return 0;
}
```

P2344.跳马问题1

Description:

Question:

> 马按中国象棋的规则走，中国象棋半张棋盘如图1所示，棋盘大小为n*m。马自左下角 (0, 0) 向右上角 (n, m) 跳。规定只能往右跳，不准往左跳，不能跳出棋盘，即x坐标只能增大。比如下图所示为一种跳行路线。
>
>  1543216994851015532
>
> 小明觉得这道题太简单，所以在棋盘上放了K个己方棋子，马不能跳到己方棋子上，**己方棋子不会绊马脚**，保证起点和终点上没有棋子，问此时跳马的路径总数。

Input:

> 第一个三个整数n, m, K（1< n, m<=15, 0<=K<n*m)
>
> 接下来K行每行两个整数，分别表示第k个己方棋子的横坐标和纵坐标

Output:

> 只有一个数：总方案数total。

Sample input:

```
样例1：
8 4 0

样例2：
4 4 1
2 0
```

Sample output:

```
样例1：
37

样例2：
1
```

Note:

> 对于4*4的棋盘原本有两条路线:
>
> (0,0)->(1,2)->(2,0)->(3,2)->(4,4)
>
> (0,0)->(1,2)->(2,4)->(3,2)->(4,4)
>
> 第一条线路经过(2, 0)，而此位置已经有棋子，所以不能走

---

src:

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

void step(int n, int m, int curX, int curY, int &total, bool **unStep)
{
    if (curX == n && curY == m)
    {
        total++;
        return;
    }
    int dir[4][2] = {{1, 2}, {2, 1}, {2, -1}, {1, -2}};
    for (int i = 0; i < 4; i++)
    {
        if (curX + dir[i][0] > n || curY + dir[i][1] > m || curY + dir[i][1] < 0)
            continue;
        if (unStep[curX + dir[i][0]][curY + dir[i][1]])
            continue;
        curX += dir[i][0];
        curY += dir[i][1];
        step(n, m, curX, curY, total, unStep);
        curX -= dir[i][0];
        curY -= dir[i][1];
    }
}

int main()
{
    int n = 0, m = 0, k = 0;
```

```
29        cin >> n >> m >> k;
30        bool **unStep = new bool *[n + 1]();
31        for (int i = 0; i < n + 1; i++)
32            unStep[i] = new bool[m + 1]();
33        for (int i = 0; i < k; i++)
34        {
35            int tmp1 = 0, tmp2 = 0;
36            cin >> tmp1 >> tmp2;
37            unStep[tmp1][tmp2] = true;
38        }
39        int total = 0;
40        step(n, m, 0, 0, total, unStep);
41        cout << total << endl;
42        return 0;
43    }
```

Week_13

P2030.数字金字塔

Description:

Question:

> 观察下面的数字金字塔。
>
> 写一个程序来查找从最高点到底部任意处结束的路径，使路径经过数字的和最大。每一步可以走到左下方的点也可以到达右下方的点。
>
> ```
> 1          7
> 2        3   8
> 3      8   1   0
> 4    2   7   4   4
> 5  4   5   2   6   5
> ```
>
> 在上面的样例中,从7 到 3 到 8 到 7 到 5 的路径产生了最大的和。

Input:

> 输入有多行。
>
> 第一个行包含 R(1<= R<=1000) ,表示行的数目。
>
> 后面每行为这个数字金字塔特定行包含的整数。
>
> 所有的被供应的整数是非负的且不大于100。

Output:

> 输出仅有一行,包含那个可能得到的最大的和。

Sample input:

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

Sample output:

```
30
```

---

src:

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

#define MAX(a, b) ((a > b) ? a : b)
#define SIZE 1000

int pyramid[SIZE][SIZE] = {0}, maxVal[SIZE][SIZE] = {0};

void way(int n, int x, int y, int pyramid[SIZE][SIZE], int maxVal[SIZE][SIZE])
{
    if (x > n)
        return;
    if (x == 1)
        maxVal[0][0] = pyramid[0][0];
    else if (y == 1)
        maxVal[x - 1][y - 1] = maxVal[x - 2][y - 1] + pyramid[x - 1][y - 1];
    else if (y == n)
        maxVal[x - 1][y - 1] = maxVal[x - 2][y - 2] + pyramid[x - 1][y - 1];
    else
        maxVal[x - 1][y - 1] = MAX(maxVal[x - 2][y - 1], maxVal[x - 2][y - 2]) +
pyramid[x - 1][y - 1];
}

int main()
{
    int n = 0;
    cin >> n;
    for (int i = 0; i < n; i++)
        for (int j = 0; j <= i; j++)
            cin >> pyramid[i][j];
    for (int i = 0; i < n; i++)
        for (int j = 0; j <= i; j++)
            way(n, i + 1, j + 1, pyramid, maxVal);
    int max = 0;
    for (int i = 0; i < n; i++)
        if (maxVal[n - 1][i] > max)
            max = maxVal[n - 1][i];
    cout << max << endl;
    return 0;
}
```

P2250.哼哼哈兮

Description:

Question:

> 有N种类型的短棒，第i种短棒长度为a_i，武力值为b_i，数量无限。这些短棒可以多个粘到一起成为长一些的棒。普通情况下，长度为A的棒和长度为B的棒组成一个长度为A+B的棒，此时
>
> - 长度为A+B的棒的武力值 = 长度为A的棒的武力值 + 长度为B的棒的武力值
>
> 但是，如果A与B的长度相等，则其武力值还要再增加233（因为这样它比较像双节棍，然后忍不住哼哼哈兮，引得大家233）。也就是说，若用两个长度为K的棒，组成一个长度为2K的棒，此时，
>
> - 长度为2K的棒的武力值 = 长度为K的棒的武力值*2 + 233。
>
> 现在，已知需要一根长M的棒，且要求武力值最大化。那么请问这根棒最大的武力值是多少？

Input:

> 第一行输入两个数N，M。
> 第二行为N个整数a_i，表示每种短棒的长度。
> 第三行为N个整数b_i，表示每种短棒的武力值。
> 输入数据保证有解。
> (0 < N, M, a_i ≤ 1000)
> (a_i ≤ M)
> (0 < b_i ≤ 10000)

Output:

> 输出一个整数，表示长M的棒的最大武力值。

Sample input:

```
3 10
1 2 3
1000 3000 6000
```

Sample output:

```
19233
```

Note

> 以下为几种可能的组成方案:
>
> ((2 2) (2 2)) (2) = ((30002+233)2+233) + 3000 = 15699
> ((2 2) (2)) (2 2) = ((30002+233) + 3000) + (30002+233) = 15466
> ((2 2) 1) (1 (2 2)) = ((30002+233)+1000)2+233 = 14699
> (2 3) (2 3) = (3000+6000)2+233 = 18233
> (1 3) (3 3) = (1000+6000)+(60002+233) = 19233
>
> 容易发现，拼接的顺序，对于最后的武力值也有影响。

src:

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

#define OFFSET 233

int a[1000] = {0}, b[1000] = {0}, val[1000] = {0};

void sort(int n)
{
    for (int i = 0; i < n; i++)
        if (b[i] > val[a[i]])
            val[a[i]] = b[i];
}
int link(int len1, int len2, int val1, int val2)
{
    if (len1 == len2)
        return val1 + val2 + OFFSET;
    else
        return val1 + val2;
}
int cut(int m)
{
    for (int i = 1; i <= m; i++)
    {
        int max = 0, tmp = 0;
        for (int j = 1; j <= i; j++)
        {
            tmp = link(i - j, j, val[i - j], val[j]);
            if (tmp > max)
                max = tmp;
        }
        val[i] = max;
    }
    return val[m];
}

int main()
{
    int n = 0, m = 0;
    cin >> n >> m;
    for (int i = 0; i < n; i++)
        cin >> a[i];
    for (int i = 0; i < n; i++)
        cin >> b[i];
    sort(n);
    cout << cut(m) << endl;
    return 0;
}
```

P1284.最小差方和

Description:

Question:

> 给定n和L，以及n个整数a[i]。
>
> 要求把a[i]分成连续的若干段。对于每一段，假设其和为S，则其代价为(S-L)^2。
>
> 整个划分方案的代价定义为每一段代价之和。
>
> 求所有划分方案中的最小代价。

Input:

> 第一行两个正整数n和L。
>
> 第二行n个整数a[i]。

Output:

> 一行一个非负整数表示最小代价。

Sample input:

Sample output:

Note:

> *data size:*
>
> 60%的数据，n<=100，L<=1000;
>
> 100%的数据，n<=3000，L<=2^20-1，|a[i]|<=100000。

---

src:

```cpp
//时间复杂度有待改善
#include <iostream>
using namespace std;

#define SIZE 4000
int n = 0, l = 0;
long long a[SIZE] = {0}, cost[SIZE] = {0};

long long _cost(int i, int j)
{
    long long sum = 0;
    for (int k = i; k <= j; k++)
        sum += a[k];
    return (sum - l) * (sum - l);
}
long long min_cost()
{
    for (int i = 1; i <= n; i++)
    {
```

```
            long long min = _cost(1, i);
            for (int j = 1; j <= i; j++)
            {
                long long tmp = cost[i - j] + _cost(i - j + 1, i);
                if (tmp < min)
                    min = tmp;
            }
            cost[i] = min;
        }
        return cost[n];
    }
    int main()
    {
        cin >> n >> l;
        for (int i = 1; i <= n; i++)
            cin >> a[i];
        cout << min_cost() << endl;
        return 0;
    }
```

P1525.飞弹拦截

Description:

Question:

> 强大的kAc建立了强大的帝国，但人民深受其学霸及23文化的压迫，于是勇敢的高达决心反抗。
>
> 高达能远程发飞弹打击，他先后对着城市开了n发，第i个飞弹高度为hi。然而kAc帝国也有一门炮可以拦截高达发射的飞弹，虽然拦截成功率为100%，但由于技术问题，这门炮每次只能拦截比上一次高度相同或更高的飞弹（第一次可以拦截任意飞弹）。玛德go负责此次拦截，他刚算好拦截方案又接到了上级的任务：一些给定的飞弹由于杀伤力大必须拦截。玛德go显然已经不屑于解决这种问题了，他显然要做一些更有意义的事情。于是这问题就交给了你，在打掉指定飞弹的前提下打掉最多飞弹。

Input:

> 第一行一个数n，第二行n个数，依次表示飞来的飞弹高度。
>
> 第三行一个数m，表示必须拦截的飞弹个数，第四行m个数表示这些飞弹的编号（0到n-1），并且保证单调递增，保证存在至少一种打击方案。

Output:

> 一个数表示最多打掉的飞弹。

Sample input:

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

Sample output:

```
4
```

Note:

> *data size:*
>
> 30%的数据n<=10
>
> 70%的数据n<=100
>
> 100%的数据n<=1000，m<=100
>
> 飞弹的高度在int表示范围内

---

src:

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

int n = 0, m = 0;
int height[1005] = {0}, maxDes[1005] = {0}, must[105] = {0};
bool getMax[1005] = {0};

int max_destroy(int left, int right, int cur)
{
    if (left == right)
        return 0;
    if (!getMax[left])
    {
        int max = 0;
        for (int i = left + 1; i <= right; i++)
        {
            int num = 0;
            bool skip = false;
            for (int j = 0; j < m; j++)
                if ((must[j] > left && must[j] < i) || (must[j] > i && height[i] > height[must[j]]))
                {
                    skip = true;
                    break;
                }
            if (skip || height[i] < cur)
                continue;
            num = max_destroy(i, right, height[i]) + 1;
            if (num > max)
                max = num;
        }
        maxDes[left] = max;
        getMax[left] = true;
    }
    return maxDes[left];
}

int main()
{
    cin >> n;
    for (int i = 0; i < n; i++)
```

```
41          cin >> height[i];
42      cin >> m;
43      for (int i = 0; i < m; i++)
44          cin >> must[i];
45      cout << max_destroy(-1, n - 1, 0) << endl;
46      return 0;
47  }
```

Week_14

P1316.最长公共子序列

Description:

Question:

> 给定两个字符串，寻找这两个字串之间的最长公共子序列。

Input:

> 输入两行，分别包含一个字符串，仅含有小写字母。

Output:

> 最长公共子序列的长度。

Sample input:

```
abcdgh

aedfhb
```

Sample output:

```
3
```

note:

> *data size*:
> 字串长度1~1000。

---

src:

```
#include <iostream>
#include <cstring>
using namespace std;

#define MAX(a, b) ((a > b) ? a : b)
#define MAXLEN 1005

char str1[MAXLEN] = {0}, str2[MAXLEN] = {0};
```

```cpp
int subLen[MAXLEN][MAXLEN] = {0};

int main()
{
    cin >> str1 >> str2;
    for (int i = 1; i <= strlen(str1); i++)
        for (int j = 1; j <= strlen(str2); j++)
        {
            if (str1[i - 1] == str2[j - 1])
                subLen[i][j] = subLen[i - 1][j - 1] + 1;
            else
                subLen[i][j] = MAX(subLen[i - 1][j], subLen[i][j - 1]);
        }
    cout << subLen[strlen(str1)][strlen(str2)] << endl;
    return 0;
}
```

P1622.采药

Description:

Question:

> 辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”

Input:

> 第一行有两个整数T（1 <= T <= 1000）和M（1 <= M <= 100），用一个空格隔开，T代表总共能够用来采药的时间，M代表山洞里的草药的数目。接下来的M行每行包括两个在1到100之间（包括1和100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

Output:

> 包括一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值。

Sample input:

```
70 3

71 100

69 1

1 2
```

Sample output:

```
3
```

Note:

> *data size*:
>
> 10;
>
> 对于全部的数据，M <= 100。

---

src:

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

#define MAX(a, b) ((a > b) ? a : b)

int T = 0, M = 0;
int cost[105] = {0}, val[105] = {0};
int maxVal[105][1005] = {0};

int main()
{
    cin >> T >> M;
    for (int i = 1; i <= M; i++)
        cin >> cost[i] >> val[i];
    for (int i = 1; i <= M; i++)
        for (int j = 1; j <= T; j++)
        {
            if (cost[i] <= j)
                maxVal[i][j] = MAX(maxVal[i - 1][j], maxVal[i - 1][j - cost[i]] + val[i]);
            else
                maxVal[i][j] = maxVal[i - 1][j];
        }
    cout << maxVal[M][T];
    return 0;
}
```

P2349.小羊与小狼

Description:

Question:

> 有n只羊与n只狼，牧羊人须要把他们摆渡到河对岸。船上最多可以容纳m只动物，任何时刻，任何地点（两岸或者船上）只要羊的数目小于狼的数目，狼就会把羊吃掉。由于牧羊人害怕寂寞，在摆渡时船上必须至少有一只动物。在船行驶到对岸之后，先所有动物瞬时下船，接着下一波动物瞬时上船。问最少摆渡次数。

Input:

> 有n只羊与n只狼，牧羊人须要把他们摆渡到河对岸。船上最多可以容纳m只动物，任何时刻，任何地点（两岸或者船上）只要羊的数目小于狼的数目，狼就会把羊吃掉。由于牧羊人害怕寂寞，在摆渡时船上必须至少有一只动物。在船行驶到对岸之后，先所有动物瞬时下船，接着下一波动物瞬时上船。问最少摆渡次数。

Output:

> 一个整数，表示答案。若无法完成，则输出-1。

Sample input:

```
3 2
```

Sample output:

```
11
```

---

src:

```
#include <iostream>
#include <limits.h>
using namespace std;

struct state
{
    int sheep;
    int wolf;
};

struct stateList
{
    state *elem;
    int num;
};

stateList get_strategy(stateList &s, int m)
{
    s.num = 0;
    for (int sheep = 0; sheep <= m; sheep++)
        for (int wolf = 0; sheep + wolf <= m; wolf++)
            if (sheep + wolf != 0 && (sheep >= wolf || sheep == 0))
            {
                s.elem[s.num].sheep = sheep;
                s.elem[s.num].wolf = wolf;
                s.num++;
            }
    return s;
}

int n = 0, m = 0, minStep = INT_MAX;
stateList strategy;
stateList memo;

state next_state(state cur, state elem, int dir)
{
```

```
    state newState = {cur.sheep + dir * elem.sheep, cur.wolf + dir * elem.wolf};
    return newState;
}

bool equal(state s1, state s2)
{
    if (s1.sheep == s2.sheep && s1.wolf == s2.wolf)
        return true;
    return false;
}

bool valid(state cur, int step)
{
    if (cur.sheep > n || cur.wolf > n || cur.sheep < 0 || cur.wolf < 0)
        return false;
    if (!(cur.sheep == 0 || cur.sheep == n || cur.sheep == cur.wolf))
        return false;
    for (int i = step - 2; i >= 0; i -= 2)
        if (equal(cur, memo.elem[i]))
            return false;
    return true;
}

void get_step(state cur, int step)
{
    int dir = ((step % 2) ? 1 : -1);
    if (cur.sheep == 0 && cur.wolf == 0)
        if (step < minStep)
        {
            minStep = step;
            return;
        }
    for (int i = 0; i < strategy.num; i++)
    {
        state newState = next_state(cur, strategy.elem[i], dir);
        if (!valid(newState, step + 1))
            continue;
        memo.elem[step + 1].sheep = newState.sheep;
        memo.elem[step + 1].wolf = newState.wolf;
        get_step(newState, step + 1);
    }
}

int main()
{
    cin >> n >> m;
    strategy.elem = new state[100000];
    get_strategy(strategy, m);
    memo.elem = new state[100000];
    memo.num = 0;
    state begin = {n, n};
    memo.elem[0] = begin;
    get_step(begin, 0);
    cout << ((minStep == INT_MAX) ? -1 : minStep) << endl;
    return 0;
}
```

P2347.过桥问题【多步决策、穷举】

Description:

Question:

> 在漆黑的夜里，n位旅行者来到了一座狭窄而且没有护栏的桥边。如果不借助手电筒的话，大家是无论如何也不敢过桥去的。不幸的是，n个人一共只带了一只手电筒，而桥窄得只够让两个人同时通过。如果各自单独过桥的话，第i个人所需要的时间是time[i]；而如果两人同时过桥，所需要的时间就是走得比较慢的那个人单独行动时所需的时间。 此时，请问让所有人都过桥，最少需要多长时间?

Input:

> 第一行包含一个整数，代表n。(n <= 6)
>
> 之后有n行，每行包含一个整数， 代表time[i]。 (time[i] <= 10000)

Output:

> 一个整数，表示最少的花费时间。

Sample input:

```
5
1
3
6
8
12
```

Sample output:

```
29
```

Note:

> 请使用课上讲解的【多步决策】方法解决。

src:

```
#include <iostream>
#include <limits.h>
using namespace std;

#define MAX(a, b) ((a > b) ? a : b)

int n = 0, cost[10] = {0}, minTime = INT_MAX, strategy[2][28] = {0}, sNum = 0;

struct state
{
    bool opp[10];
} memo[1000] = {0};

```

```
void get_strategy()
{
    for (int i = 1; i <= n; i++)
        for (int j = i; j <= n; j++)
        {
            strategy[0][sNum] = i;
            strategy[1][sNum] = j;
            sNum++;
        }
}
bool done(state cur)
{
    for (int i = 1; i <= n; i++)
        if (!cur.opp[i])
            return false;
    return true;
}
state next_state(state cur, int k, int dir)
{
    state newState = cur;
    newState.opp[strategy[0][k]] = (dir ? true : false);
    newState.opp[strategy[1][k]] = (dir ? true : false);
    return newState;
}
bool equal(state s1, state s2)
{
    for (int i = 1; i <= n; i++)
        if (s1.opp[i] != s2.opp[i])
            return false;
    return true;
}
bool valid(state cur, int k, bool dir, int step)
{
    if (dir)
        if (cur.opp[strategy[0][k]] || cur.opp[strategy[1][k]])
            return false;
    if (!dir)
        if ((!cur.opp[strategy[0][k]]) || (!cur.opp[strategy[1][k]]))
            return false;
    for (int i = step - 2; i >= 0; i--)
        if (equal(next_state(cur, k, dir), memo[i]))
            return false;
    return true;
}
void login(state s, int step)
{
    for (int i = 1; i <= n; i++)
        memo[step].opp[i] = s.opp[i];
}
bool judge(state cur, int k, int dir)
{
    int num = 0;
    for (int i = 1; i <= n; i++)
        if (!cur.opp[i])
            num++;
    if (dir && num != 1)
        if (strategy[0][k] == strategy[1][k])
            return false;
```

```
72        if (!dir)
73            if (strategy[0][k] != strategy[1][k])
74                return false;
75        return true;
76    }
77    void get_step(state cur, int step, int curTime)
78    {
79        bool dir = ((step % 2) ? false : true);
80        if (done(cur))
81        {
82            if (curTime < minTime)
83                minTime = curTime;
84            return;
85        }
86        for (int i = 0; i < sNum; i++)
87        {
88            if (!valid(cur, i, dir, step + 1))
89                continue;
90            if (!judge(cur, i, dir))
91                continue;
92            state nextState = next_state(cur, i, dir);
93            login(nextState, step + 1);
94            get_step(nextState, step + 1, curTime + MAX(cost[strategy[0][i]],
    cost[strategy[1][i]]));
95        }
96    }
97    int main()
98    {
99        cin >> n;
100       for (int i = 1; i <= n; i++)
101           cin >> cost[i];
102       get_strategy();
103       memo[0] = {0};
104       state start = {0};
105       get_step(start, 0, 0);
106       cout << minTime << endl;
107       return 0;
108   }
```

Final_Practice

P2200. 函数计算

Description:

Question:

> 有一个数列，f(n) = f(n-1) + f(n//2) + f(n//3)，其中n//m表示整除。 如 3//2=1, 4//2=2, 5//2=2。
>
> 当n<=0时，有f(n) = 0。
>
> 现在告诉你f(1)=x，请计算f(n)的值.

> 请使用递归完成计算。

Input:

> 输入两个整数x、n。
>
> 0 < x，n <= 20

Output:

> 输出f(n)的值。

Sample input:

```
1 6
```

Sample output:

```
16
```

Notes:

> 样例含义
>
> f(1) = 1
>
> f(2) = f(1) + f(1) + f(0) = 1+1+0 = 2
>
> f(3) = f(2) + f(1) + f(1) = 2+1+1 = 4
>
> f(4) = f(3) + f(2) + f(1) = 4+2+1 = 7
>
> f(5) = f(4) + f(2) + f(1) = 7+2+1 = 10
>
> f(6) = f(5) + f(3) + f(2) = 9+4+2 = 16

Src:

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

int fun(int n, int x)
{
    if (n == 0)
        return 0;
    else if (n == 1)
        return x;
    else
        return fun(n - 1, x) + fun(n / 2, x) + fun(n / 3, x);
}

int main()
{
    int x = 0, n = 0;
    cin >> x >> n;
```

```
18        cout << fun(n, x) << endl;
19        return 0;
20    }
```

P2201.灰化肥与黑化肥

Description:

Question:

> 小明有个好朋友，她叫花非花。今天花非花有n克灰化肥，m克黑化肥。黑化肥挥发会发灰，变成灰化肥。灰化肥挥发会发黑，变成黑化肥。花非花的灰化肥只有白天会挥发，花非花的黑化肥只有晚上会挥发，一整个白天灰化肥会挥发x%，一整个晚上黑化肥会挥发y%。问过了t天，花非花的灰化肥和黑化肥各多少克?

Input:

> 输入包含五个整数：n、m、x、y、t，含义如题目所述。

Output:

> 每组数据，输出第t天灰化肥和黑化肥的克数。（小数点后保留4位）

Sample input:

```
10 8 25 60 1
```

Sample output:

```
13.8000 4.2000
```

Notes:

> 【数据范围】
>
> 对于所有的数据：0 <= n、m、x、y <= 100
>
> 对于80%的数据：0 <= t <= 100
>
> 对于100%的数据：0 <= t <= 10^9
>
> 【第一组样例解释】
>
> 最开始有10克灰化肥，8克黑化肥。第一天白天有10 *0.25=2.5*克灰化肥会挥发变成黑化肥，因此经过白天后，有*7.5*克灰化肥和*10.5*克黑化肥。第一天晚上有*10.5* 0.60=6.3克黑化肥会挥发变成灰化肥。因此，经过一天以后，花非花有13.8克灰化肥和4.2克黑化肥。
>
> 【增加一组样例】
>
> Sample Input:
>
> 30 20 70 22 5

Sample Output:

14.3713 35.6287

【小数点后保留4位的示例代码】

#include // 需要包含此头文件

#include

using namespace std;

int main() {

double ans = 123.456789;

cout << setiosflags(ios::fixed) << setprecision(4) << ans <<endl;

return 0;

}

Src:

```
//时间复杂度有待改善
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float gray = 0, black = 0, x = 0, y = 0;
    int t = 0;
    cin >> gray >> black >> x >> y >> t;
    int sum = gray + black;
    x = x / 100;
    y = y / 100;
    for (int i = 0; i < t; i++)
    {
        float curG = gray, curB = black;
        gray = curG * (1 - x + x * y) + curB * y;
        black = sum - gray;
    }
    cout << fixed << setprecision(4) << gray << ' ' << black << endl;
    return 0;
}
```

P2202.程序帮我写论文

Description:

Question:

> 小明最近在赶论文，其中，论文大小标题的编号让他十分困扰。比如他写完的初稿长这样：
>
> \1. Introduction
>
> \2. Proposed Method
>
> 2.1. –
>
> 2.2. O.O
>
> 2.3. QwQ
>
> \3. Experiments
>
> \4. Conclusions
>
> 然后他把稿子给徐老师看过以后，徐老师说2.1节不要了！于是小明删除了2.1节，然后就变成了这个样子：
>
> \1. Introduction
>
> \2. Proposed Method
>
> 2.2. O.O
>
> 2.3. QwQ
>
> \3. Experiments
>
> \4. Conclusions
>
> 这时候发现，论文小标题的编号就不正确了！于是他又要重新给小标题编号。这让他非常苦恼。
>
> 于是他想，能不能让程序自动给小标题编号呢？
>
> 为了完成这一设想，他决定先做一个能够正确输出编号的程序。
>
> 为了简单起见，我们规定在论文中，一级标题有x个，每个标题下有x个子标题，整篇一共有y级标题。
>
> 其中，"1."，"2." 为1级标题， "1.1.", "1.2.", "2.2." 等都为二级标题。以此类推。
>
> 请注意每级标题末尾的"."。

> 编号结束后，跟随一个空格，并加上标题名称。为了简单起见，所有标题名称都为"Title"
>
> 你能帮帮小明么?

Input:

> 输入包含两个整数：x、y。
>
> 0 < x, y < 10

Output:

> 输出这篇论文

Sample input:

```
2 3
```

Sample output:

```
1. Title
1.1. Title
1.1.1. Title
1.1.2. Title
1.2. Title
1.2.1. Title
1.2.2. Title
2. Title
2.1. Title
2.1.1. Title
2.1.2. Title
2.2. Title
2.2.1. Title
2.2.2. Title
```

Notes:

> 【样例解释】
>
> 输入说明，每个标题所含子标题个数为2，所以最包含"1."和"2."两个标题。
>
> 对于"1."这个标题，其包含"1.1"和"1.2"这两个子标题。
>
> 同样的，对于"1.1."标题，其包含"1.1.1."和"1.1.2."两个子标题。
>
> 由于输入说明最大标题级数为3，所以标题"1.1.1."和"1.1.2."不再包含子标题。

Src:

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

void write(int x, int y, int cur, int *a)
```

```
{
    if (cur > y)
        return;
    for (int i = 0; i < x; i++)
    {
        a[cur - 1] = i + 1;
        for (int j = 0; j < cur; j++)
            cout << a[j] << '.';
        cout << ' ' << "Title" << endl;
        write(x, y, cur + 1, a);
    }
}

int main()
{
    int x = 0, y = 0;
    cin >> x >> y;
    int a[100] = {0};
    write(x, y, 1, a);
    return 0;
}
```

P2202.Markdown

Description:

Question:

> Markdown是一个简单的标记语言。不像Word，Markdown完全用纯文本的方式，来表达内容和格式。
>
> 其支持基本的常用格式，如标题、加粗、斜体、列表等，能很方便的应对日常写作的需求。
>
> 那么今天，我们就来实现一个简化版本的Markdown编译器！
>
> 简化的Markdown包含两类格式：
>
> \1. 标题与段落格式。所有以'#'开头的行，被认为是标题行。
>
> 若以'# '开头，则认为是一级标题，转换后，需要在标题文字前后"
>
> ""
>
> ", 如:
>
> "# Title abc" --> "
>
> # Title abc
>
> "
>
> 若以'## '开头，则认为是二级标题，转换后，需要在标题文字前后"

"""

", 如:

"## Title abc" --> "

Title abc

"

若以'### '开头，则认为是三级标题，转换后，需要在标题文字前后"

"""

", 如:

"### Title abc" --> "

Title abc

"

最高只有三级标题。

除此以外，其他行被认为是段落行。转换后，需要在标题文字前后"

""

", 如:

""good morning" --> "

good morning

""

特别注意首个空格问题，即

正确示例："### Title abc" --> "

Title abc

"

错误示例："### Title abc" --> "

Title abc

"

保证所有的标题，最后一个#和Title之间都有一个空格

\2. 字体格式

所有的字符都可以有以下三种格式:

斜体：用'*'包含的字符被认为是斜体，转换后需要用" ”” "包裹，如:

"*good *" --> "

good

"

加粗：用'**'包含的字符被认为是加粗，转换后需要用"""包裹，如：

"**good **" --> "

good

"

加粗斜体：用'***'包含的字符被认为是加粗斜体，转换后需要用" ”” "包裹，如：

"***good ***" --> "

good

"

所有的字体格式不会嵌套，不会跨行。正文内容中保证不存在' '，所有的' '均表示字体格式。

为了简化问题，规定标题中不存在字体格式。文件中的所有空行将被忽略。不存在连续两个空格。

Input:

从文件input.txt读入。内容为用简化Markdown格式书写的文件，每行不超过1000个字符。

保证输入数据符合题目要求。

注意：每行的结尾可能为'\n'，也可能为'\r' '\n'。请注意**去除**读入后**每行末尾的所有'\r', '\n'**。输入文件**最后一行**可能**没有**换行符。

Output:

将转换后的文本，输出到文件output.txt。每行的结尾限定为'\n'。

Sample input:

```
# good morning

what a **beautiful** day! *wow*

## lalala

one ***line ***q***w***q

another line qwq
```

Sample output:

```
<h1>good morning</h1>
<p>what a <strong>beautiful</strong> day! <em>wow</em></p>
<h2>lalala</h2>
<p>one <strong><em>line </em></strong>q<strong><em>w</em></strong>q</p>
<p>another line qwq</p>
```

Notes:

> 【数据组成说明】
>
> 80%的数据，仅包含标题与段落格式，不存在字体格式。
>
> 10%的数据，字体格式中只有斜体，不包含加粗和加粗斜体。
>
> 10%的数据，包含上述所有格式。
>
> 【增加一组针对80%数据的样例】
>
> Sample Input
>
> # Title1
> lala
> ## Title2
> haha
> ### Title3
> hehe
>
> Sample Output
>
> <h1>Title1</h1>
> lala
> <h2>Title2</h2>
> haha
> <h3>Title3</h3>
> hehe

Src:

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

#define MAX 10000

void convert(char *inStr, ofstream &out)
{
    if (strlen(inStr) == 0)
        return;
    if (inStr[0] == '#')
    {
        char outStr[MAX] = {0};
        int num = 1;
        for (int i = 1; inStr[i] == '#'; i++)
            num++;
        char head[10] = "<h >", tail[10] = "</h >";
        head[2] = '0' + num;
        tail[3] = '0' + num;
```

```
            for (int i = num + 1, j = 0; inStr[i] && (inStr[i] != '\n') && (inStr[i] != '\r'); i++, j++)
                outStr[j] = inStr[i];
            out << head << outStr << tail;
        }
        else
        {
            char head[10] = "<p>", tail[10] = "</p>", word[MAX] = {0};
            bool bold = false, italic = false;
            out << head;
            for (int i = 0, j = 0;; i++, j++)
            {
                if (!(inStr[i] && (inStr[i] != '\n') && (inStr[i] != '\r')))
                {
                    out << word;
                    break;
                }
                if (inStr[i] == ' ')
                {
                    word[j] = inStr[i];
                    if (!(bold && italic))
                    {
                        out << word;
                        memset(word, 0, sizeof(word));
                        j = -1;
                    }
                }
                else if (inStr[i] == '*')
                {
                    if (bold || italic)
                    {
                        if (bold)
                            out << "<strong>";
                        if (italic)
                            out << "<em>";
                        out << word;
                        if (italic)
                            out << "</em>";
                        if (bold)
                            out << "</strong>";
                        if (bold)
                        {
                            if (italic)
                                i += 2;
                            else
                                i += 1;
                        }
                        memset(word, 0, sizeof(word));
                        bold = false;
                        italic = false;
                        j = -1;
                    }
                    else
                    {
                        if (j != 0)
                        {
                            out << word;
                            j = -1;
```

```
                    }
                    int k = 1;
                    for (; k < 3; k++)
                        if (inStr[i + k] != '*')
                            break;
                    if (k > 1)
                        bold = true;
                    if (k != 2)
                        italic = true;
                    i += k - 1;
                    if (j != -1)
                        j--;
                }
            }
            else
                word[j] = inStr[i];
        }
        out << tail;
    }
    out << endl;
}

int main()
{
    ifstream in("input.txt");
    ofstream out("output.txt");
    if (!(in && out))
        return -1;
    else
    {
        char inStr[MAX] = {0}, outStr[MAX] = {0};
        while (!in.eof())
        {
            memset(inStr, 0, sizeof(inStr));
            memset(outStr, 0, sizeof(outStr));
            in.getline(inStr, MAX);
            convert(inStr, out);
        }
    }
    return 0;
}

```

P2199.老实人和骗子

Description:

Question:

> 四个人中有两个是骗子，另外两个是老实人。骗子一直说假话，老实人一直说真话。现在所有人都指出了其中一个人的身份，问这四个人所有可能的身份情况有几种?

Input:

> 输入有4行，分别表示四个人所说的话。

> 第i行表示第i个人说的话，包含两个整数 j, k，分别代表所指向人的编号，以及其身份。其中，k=1 表示其为老实人，k=0表示其为骗子。
>
> 1 <= i,j <= 4

Output:

> 输出所有可能的情况数。

Sample input:

```
1 1
1 1
1 0
1 0
```

Sample output:

```
2
```

Notes:

> 【样例解释】
>
> 输入数据代表一下场景：
>
> 1号：1号是老实人
>
> 2号：1号是老实人
>
> 3号：1号是骗子
>
> 4号：1号是骗子
>
> 仅存在以下两种情况：
>
> 情况一：1,2为老实人，3,4为骗子。
>
> 情况二：1,2为骗子，3,4为老实人。
>
> 注意：“1、2为老实人，3、4为骗子”与“2、1为老师人，3、4为骗子”是同一种情况。

Src:

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

int main()
{
    int man[5] = {0};
    int num = 0;
    int word[5][2] = {0};
    for (int i = 1; i <= 4; i++)
```

```
        cin >> word[i][0] >> word[i][1];
    for (man[1] = 0; man[1] < 2; man[1]++)
        for (man[2] = 0; man[2] < 2; man[2]++)
            for (man[3] = 0; man[3] < 2; man[3]++)
                for (man[4] = 0; man[4] < 2; man[4]++)
                {
                    int sum = 0;
                    for (int i = 1; i <= 4; i++)
                        sum += man[i];
                    if (sum != 2)
                        continue;
                    for (int i = 1; i <= 5; i++)
                    {
                        if (i == 5)
                            num++;
                        else
                        {
                            if (man[i] == 1)
                                if (man[word[i][0]] != word[i][1])
                                    break;
                            if (man[i] == 0)
                                if (man[word[i][0]] == word[i][1])
                                    break;
                        }
                    }
                }
    cout << num << endl;
    return 0;
}
```

P2246.扑克牌游戏

Description:

Question:

> 扑克牌有4种花色：黑桃♠（Spade）、红心♥（Heart）、梅花♣（Club）、方块♦（Diamond）。每种花色有13张牌，编号从小到大为：A,2,3,4,5,6,7,8,9,10,J,Q,K。
>
> 对于一个扑克牌堆，定义以下4种操作命令：
>
> \1) 添加（Append）：添加一张扑克牌到牌堆的底部。如命令“Append Club Q”表示添加一张梅花Q到牌堆的底部。
>
> \2) 抽取（Extract）：从牌堆中抽取某种花色的所有牌，按照编号从小到大进行排序，并放到牌堆的顶部。如命令“Extract Heart”表示抽取所有红心牌，排序之后放到牌堆的顶部。
>
> \3) 反转（Revert）：使整个牌堆逆序。
>
> \4) 弹出（Pop）：如果牌堆非空，则除去牌堆顶部的第一张牌；如果牌堆为空，则不进行操作。
>
> 初始时牌堆为空。输入n个操作命令（1 ≤ n ≤1000），请问最终牌堆的顶部是什么牌？
>
> 注意：每种花色和编号的牌数量不限。

Input:

> 第一行输入一个整数n，表示命令的数量。

> 接下来的n行，每一行输入一个命令。

Output:

> 输出共1行。如果最终牌堆为空，输出“null”；如果最终牌堆非空，输出牌堆顶部的牌的花色和编号（字母或数字），用空格分隔，并注意字母的大小写。

Sample input:

```
6
Append Club Q
Append Diamond 5
Append Club 10
Extract Club
Revert
Pop
```

Sample output:

```
Club Q
```

Notes:

> 无

Src:

```
1
```

P2247.考试成绩可视化

Description:

Question:

> 每次考试过后，老师都要看下大家考的怎么样。因此，需要绘制一个分数分布直方图。
>
> 现在需要在命令行中输出这个分布图。我们将考生分数按10分为阶梯，分成10个区间段，分别是[0, 10), [10, 20), …, [80, 90), [90, 100]。其中[a,b)表示**大于等于a，小于b**的区间（特别注意100包含在最后一个区间中）。 而后，我们统计每个分数段中学生的个数，以分数段为纵轴，以每个分数段中的学生个数为横轴，就可以画出分数分布直方图了。
>
> 为了简化问题，规定分布直方图的大小为10*10，横轴每格表示1人。当某个分数段有X人时，若X<=10，则横轴长度为X；若X>10，横轴长度为10。（注意：此处的直方图是横着画的，与常见的直方图有所不同）
>
> 现给出一次考试中N个学生的分数，请你画出这场考试的分数分布直方图。

Input:

> 第一行一个整数N。
> 第二行有N个整数a_i，表示每个学生的分数。
> 0 < N <= 100, 0 <= a_i <= 100

Output:

> 输出考试分数分布直方图。直方图大小为10*10，其中’ . ’表示空白，’@’表示非空白。

Sample input:

```
36
100 99 98 98 90 95 80 81 82 83 84 85 86 87 88 89 89 70 71 72 73 74 75 76 60 64 62
63 64 55 51 52 42 42 30 2
```

Sample output:

```
@.........
..........
..........
@.........
@@........
@@@.......
@@@@@.....
@@@@@@@...
@@@@@@@@@@
@@@@@@....
```

Notes:

> 无

Src:

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

int main()
{
    int n = 0, tmp = 0;
    cin >> n;
    int score[10] = {0};
    for (int i = 0; i < n; i++)
    {
        cin >> tmp;
        for (int j = 0; j < 10; j++)
            if (tmp >= 10 * j && tmp < 10 * (j + 1))
            {
                score[j]++;
                break;
            }
        if (tmp == 100)
            score[9]++;
    }
    for (int i = 0; i < 10; i++)
    {
        if (score[i] > 10)
            score[i] = 10;
        for (int j = 0; j < score[i]; j++)
            cout << '@';
        for (int j = 10 - score[i]; j > 0; j--)
```

```
            cout << '.';
        cout << endl;
    }
    return 0;
}
```

P2248.超级逻辑推理

Description:

Question:

> 一机密研究所中一共有五个研究人员。最近，A探长收到情报，这五个人中混入了两个间谍。于是A探长分别将五个人员隔离开来一一问话，每个人都说出了其中一个人的身份。已知间谍一定说假话，其他人一定说真话。那么，五个人所有可能的身份情况有几种?

Input:

> 输入有5行，分别表示五个人所说的话。
> 第i行表示第i个人说的话，包含两个整数 j, k，分别代表所指向人的编号，以及其身份。其中，k=1表示其为好人，k=0表示其为间谍。
> (1 <= i ,j <= 5)

Output:

> 输出所有可能的身份情况数。

Sample input:

```
1 1
1 1
1 0
1 0
5 1
```

Sample output:

```
2
```

Notes:

> 无

Src:

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

int main()
{
    int man[6] = {0};
    int num = 0;
    int word[6][2] = {0};
    for (int i = 1; i <= 5; i++)
```

```
        cin >> word[i][0] >> word[i][1];
    for (man[1] = 0; man[1] < 2; man[1]++)
        for (man[2] = 0; man[2] < 2; man[2]++)
            for (man[3] = 0; man[3] < 2; man[3]++)
                for (man[4] = 0; man[4] < 2; man[4]++)
                    for (man[5] = 0; man[5] < 2; man[5]++)
                    {
                        int sum = 0;
                        for (int i = 1; i <= 5; i++)
                            sum += man[i];
                        if (sum != 3)
                            continue;
                        for (int i = 1; i <= 6; i++)
                        {
                            if (i == 6)
                                num++;
                            else
                            {
                                if (man[i] == 1)
                                    if (man[word[i][0]] != word[i][1])
                                        break;
                                if (man[i] == 0)
                                    if (man[word[i][0]] == word[i][1])
                                        break;
                            }
                        }
                    }
    cout << num << endl;
    return 0;
}
```

P2249.明明的幸运数

Description:

Question:

> 明明喜欢7这个数字，如果一个数的十进制表示中含有7，则该数为明明的幸运数。例如7，78，17都是幸运数，而168不是幸运数。
>
> 如果一个数能够被幸运数整除，则称之为近似幸运数。例如14能被7整除，是一个近似幸运数。
>
> 现给定两个正整数，请输出这个区间内（含区间首尾）既不是幸运数也不是近似幸运数的那些数的数量；如果不存在，则输出0。

Input:

> 输入两个整数a, b。

Output:

> 输出为一个数，表示[a, b]区间中既不是幸运数也不是近似幸运数的数的数量。

Sample input:

```
1 10
```

Sample output:

```
9
```

Notes:

> 【数据范围】
>
> 50%的数据满足：1 <= a,b <= 10,000
>
> 100%的数据满足： 1 <= a,b <= 1,000,000

Src:

```cpp
//时间复杂度有待改善
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <cstring>
using namespace std;

int main()
{
    int a = 0, b = 0, cur = 0, num = 0;
    cin >> a >> b;
    bool *luck = new bool[b]();
    for (int i = 0; i < b; i++)
    {
        char str[100] = {0};
        sprintf(str, "%d", i + 1);
        for (int j = 0; j < strlen(str); j++)
            if (str[j] == '7')
            {
                luck[i] = true;
                break;
            }
    }
    for (cur = a; cur <= b; cur++)
    {
        bool lucky = false;
        if (luck[cur - 1])
            continue;
        for (int i = 1; i < cur; i++)
            if (luck[i - 1])
                if (cur % i == 0)
                {
                    lucky = true;
                    break;
                }
        if (!lucky)
            num++;
    }
    cout << num << endl;
    return 0;
}
```

P2250.哼哼哈兮

Description:

Question:

> 有N种类型的短棒，第i种短棒长度为a_i，武力值为b_i，数量无限。这些短棒可以多个粘到一起成为长一些的棒。普通情况下，长度为A的棒和长度为B的棒组成一个长度为A+B的棒，此时
>
> - 长度为A+B的棒的武力值 = 长度为A的棒的武力值 + 长度为B的棒的武力值
>
> 但是，如果A与B的长度相等，则其武力值还要再增加233（因为这样它比较像双节棍，然后忍不住哼哼哈兮，引得大家233）。也就是说，若用两个长度为K的棒，组成一个长度为2K的棒，此时，
>
> - 长度为2K的棒的武力值 = 长度为K的棒的武力值*2 + 233。
>
> 现在，已知需要一根长M的棒，且要求武力值最大化。那么请问这根棒最大的武力值是多少？

Input:

> 第一行输入两个数N，M。
> 第二行为N个整数a_i，表示每种短棒的长度。
> 第三行为N个整数b_i，表示每种短棒的武力值。
> 输入数据保证有解。
> (0 < N, M, a_i ≤ 1000)
> (b_i ≤ M)
> (0 < b_i ≤ 10000)

Output:

> 输出一个整数，表示长M的棒的最大武力值。

Sample input:

```
3 10
1 2 3
1000 3000 6000
```

Sample output:

```
19233
```

Notes:

> 以下为几种可能的组成方案:
>
> ((2 2) (2 2)) (2) = ((3000 *2+233)* 2+233) + 3000 = 15699
> ((2 2) (2)) (2 2) = ((3000*2+233) + 3000) + (3000*2+233) = 15466
> ((2 2) 1) (1 (2 2)) = ((3000*2+233)+1000)*2+233 = 14699
> (2 3) (2 3) = (3000+6000)*\2+233 = 18233
> (1 3) (3 3) = (1000+6000)+(6000*2+233) = 19233
>
> 容易发现，拼接的顺序，对于最后的武力值也有影响。

Src:

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

#define OFFSET 233

int n = 0, m = 0;
int len[1005] = {0}, val[1005] = {0}, val_len[1005] = {0};

void sort()
{
    for (int i = 1; i <= n; i++)
        if (val[i] > val_len[len[i]])
            val_len[len[i]] = val[i];
}

int connect(int len1, int val1, int len2, int val2)
{
    int res = val1 + val2;
    if (len1 == len2)
        res += OFFSET;
    return res;
}

int cut(int x)
{
    for (int i = 1; i <= x; i++)
        for (int j = 0; j < i; j++)
        {
            int tmp = connect(j, val_len[j], i - j, val_len[i - j]);
            if (tmp > val_len[i])
                val_len[i] = tmp;
        }
    return val_len[x];
}

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> len[i];
    for (int i = 1; i <= n; i++)
        cin >> val[i];
    sort();
    cout << cut(m) << endl;
    return 0;
}
```

P2031.偶数个3

Description:

Question:

> 在所有的N位正整数中，有多少个数中有偶数个数字3？ （0个，也即不包含3，也计算为包含偶数个数字3）

Input:

> 输入仅有一行，为一个正整数N。（1<=N<=7）

Output:

> 输出仅有一行，为一个整数C，即方案数。

Sample input:

```
2
```

Sample output:

```
73
```

Notes:

> 无

Src:

```
1
```

P2032.占卜

Description:

Question:

> 助教喜欢打叉和占卜,然后就想出了一个以画叉占卜的方法。
>
> 给定任意一个正整数N，可以把N^2个整数按顺时针螺旋的方向填进NxN的表格中。若N个正整数依次为0, 1, ..., N-1, 则填入表格后吧
>
> N=4
> ```
> 0  1  2  3
> 11 12 13 4
> 10 15 14 5
> 9  8  7  6
> ```

```

N=5
```

```
0  1  2  3  4
15 16 17 18 5
14 23 24 19 6
13 22 21 20 7
12 11 10 9  8
```

然后在NxN的表格上画一个叉，(即两条对角线上的元素全部标成X)

N=4

```
XOOX
OXXO
OXXO
XOOX
```

N=5

```
XOOOX
OXOXO
OOXOO
OXOXO
XOOOX
```

把X对应位置上的数值加起来即当天自己的幸运数字。

Input:

输入一共有两行

第一行是一个整数N, 1 <= N <= 500

第二行是N^2个整数, 每个整数的取值范围为[0, 100], 以一个空格隔开

Output:

> 输出只有一个整数，为输入对应的幸运数字。

Sample input:

```
3
45 28 26 51 71 49 16 53 49
```

Sample output:

```
207
```

Notes:

> 无

Src:

```

```

P2033.二的幂的表示

Description:

Question:

> 任何一个正整数都可以用2进制表示，例如：137的2进制表示为10001001。
>
> 将这种2进制表示写成2的次幂的和的形式，令次幂高的排在前面，可得到如下表达式：
> 137=2^{7+2}3+2^0
>
> 现在约定幂次用括号来表示，即a^b表示为a（b）
>
> 此时，137可表示为：2(7)+2(3)+2(0)
>
> 又如：1315=2^{10+2}8+2^5+2+1
>
> 所以1315最后可表示为：2(10)+2(8)+2(5)+2(1)+2(0)

Input:

> 输入仅有一行，为一个正整数n（1<=n<=20000）

Output:

> 输出仅有一行，为符合约定的n的2的幂次的表示（在表示中不能有空格）

Sample input:

```
137
```

Sample output:

```
2(7)+2(3)+2(0)
```

Notes:

> 样例二：
>
> 【输入】
>
> 1315
>
> 【输出】
>
> 2(10)+2(8)+2(5)+2(1)+2(0)

Src:

```
1
```

P2034.选择排序（使用递归完成）

Description:

Question:

> 排序，顾名思义，是将若干个元素按其大小关系排出一个顺序。形式化描述如下：有n个元素a[1]，a[2]，…，a[n]，从小到大排序就是将它们排成一个新顺序a[i[1]]<a[i[2]]<…<a[i[n]]， i[k]为这个新顺序。
>
> 选择排序的思想极其简单，每一步都把一个最小元素放到前面，如果有多个相等的最小元素，选择排位较靠前的放到当前头部。还是那个例子：{3 1 5 4 2}：
>
> 第一步将1放到开头（第一个位置），也就是交换3和1，即swap(a[0],a[1])得到{1 3 5 4 2}
>
> 第二步将2放到第二个位置，也就是交换3和2，即swap(a[1],a[4])得到{1 2 5 4 3}
>
> 第三步将3放到第三个位置，也就是交换5和3，即swap(a[2],a[4])得到{1 2 3 4 5}
>
> 第四步将4放到第四个位置，也就是交换4和4，即swap(a[3],a[3])得到{1 2 3 4 5}
>
> 第五步将5放到第五个位置，也就是交换5和5，即swap(a[4],a[4])得到{1 2 3 4 5}
>
> 输入n个整数，输出选择排序的全过程。

Input:

> 第一行一个正整数n，表示元素个数
>
> 第二行为n个整数，以空格隔开

Output:

> 共n行，每行输出第n步选择时交换哪两个位置的下标，以及交换得到的序列，
>
> 格式如下:
>
> swap(a[i], a[j]):a[0] … a[n-1]
>
> i和j为所交换元素的下标，下标从0开始，最初元素顺序按输入顺序。另外请保证i<=j，a[0]…a[n-1]为交换后的序列，元素间以一个空格隔开。
>
> 注意：逗号后面有一个空格，冒号后面没有。

Sample input:

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

Sample output:

```
swap(a[0], a[2]):1 3 4 1 2
swap(a[1], a[3]):1 1 4 3 2
swap(a[2], a[4]):1 1 2 3 4
swap(a[3], a[3]):1 1 2 3 4
swap(a[4], a[4]):1 1 2 3 4
```

Notes:

> 【数据规模】
>
> n<=100，整数元素在int范围内

Src:

```
1
```

P1994.我要炒股！

Description:

Question:

> 有没有熔断机制都挡不住我炒股的心！ ——金百万
>
> 金百万小朋友最近迷上了炒股，梦想着一夜暴富，当上CEO，迎娶白富美~但是首先，他要先赚到钱才行。幸运的是，他获得了某只股票的连续n天内的价格走势，那么怎么操作才能使他在这n天内的收益达到最大呢？最大收益又是多少呢？金百万小朋友的智商不太够用了，只能求助你啦。
>
> 每天金百万可以执行买入/卖出操作一次（可以只买入，可以只卖出，也可以买入后卖出，也可以卖出持有股后买入）。为了简化问题，我们假设金百万只能同时持有一股。股票买卖没有交易费。
>
> 当然，金百万小朋友是不缺本金哒，总有足够的钱买入。

Input:

> 从文件stock.in输入。
>
> 输入包括两行。
>
> 第一行为一个正整数n。
>
> 第二行为n个整数，表示当天的股票价格。

Output:

> 输出到文件stock.out中
>
> 一个非负整数，表示最大盈利。

Sample input:

```
3
2 1 3
```

Sample output:

```
2
```

Notes:

> 样例解释：第二天买进，第三天卖出。
>
> 请使用long long存储盈利。
>
> n<=10000，-1000000<=每日价格<=1000000

Src:

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

int main()
{
    ifstream in("stock.in");
    ofstream out("stock.out");
    int n = 0;
    long sum = 0;
    in >> n;
    long *price = new long[n + 1]();
    for (int i = 1; i <= n; i++)
        in >> price[i];
    in.close();
    for (int i = 1; i < n; i++)
    {
        if (price[i] >= price[i + 1])
            continue;
        if (price[i] < price[i + 1])
            sum += price[i + 1] - price[i];
    }
    out << sum << endl;
    out.close();
```

```
25        return 0;
26    }
```

P1601.侦探推理2.0

Description:

Question:

> 明明同学最近迷上了侦探漫画《柯南》并沉醉于推理游戏之中，于是他召集了一群同学玩推理游戏。游戏的内容是这样的，明明的同学们先商量好由其中的一个人充当罪犯（在明明不知情的情况下），明明的任务就是找出这个罪犯。接着，明明逐个询问每一个同学，被询问者可能按照下表中限定的语法格式选择其中一种来回答：
>
> | 证词语法格式 | 证词含义 |
> | --- | --- |
> | I am guilty. | 我是罪犯 |
> | I am not guilty. | 我不是罪犯 |
> | XXX is guilty. | XXX是罪犯（XXX表示某个同学的名字） |
> | XXX is not guilty. | XXX不是罪犯 |
> | Today is XXX. | 今天是XXX（XXX表示星期几，是Monday Tuesday WednesdayThursday Friday Saturday Sunday其中之一） |
>
> 证词中出现的其他话，都不列入逻辑推理的内容。被询问者可能会被问多次，因此可能是有好几条证词是同一个人说的。
>
> 明明所知道的是，他的同学中有N个人始终说假话，其余的人始终说真。
>
> 现在，明明需要你帮助他从他同学的话中推断出谁是真正的凶手，请记住，凶手只有一个！
>
> 提示：对“今天是星期几”证言的理解比较关键，答案中不能有多人关于星期几的叙述产生矛盾。注意N代表的是始终说谎的人数，也就是说，一个人要么始终说真话，要么始终说假话。

Input:

> 输入由若干行组成，第一行有二个整数，M（1≤M≤20）、N（1≤N≤M）和P（1≤P≤100）；
>
> M是参加游戏的明明的同学数，N是其中始终说谎的人数，P是证言的总数。
>
> 接下来M行，每行是明明的一个同学的名字（英文字母组成，没有主格，全部大写）。
>
> 之后有P行，每行开始是某个同学的名字，紧跟着一个冒号和一个空格，后面是一句证词，符合前表中所列格式。证词每行不会超过250个字符。
>
> 输入中不会出现连续的两个空格，而且每行开头和结尾也没有空格。

Output:

> 如果你的程序能确定谁是罪犯，则输出他的名字；如果程序判断出不止一个人可能是
>
> 罪犯，则输出 Cannot Determine；如果程序判断出没有人可能成为罪犯，则输出 Impossible。限定大小写。

Sample input:

```
3 1 5
MIKE
CHARLES
KATE
MIKE: I am guilty.
MIKE: Today is Sunday.
CHARLES: MIKE is guilty.
KATE: I am guilty.
KATE: How are you??
```

Sample output:

```
MIKE
```

Notes:

> 无

Src:

```
1
```

P1688.小Cutey采果子

Description:

Question:

> 在一片树林中，有n棵树排成了一条直线，每棵树上可能有若干个果子，也可能有毒蛇。小Cutey在第一棵树上，每次它可以向前跳p棵树或者q棵树，但是它不能跳到有毒蛇的树上去。而它的目标是跳到第n棵树，并且采到尽量多的果子。
>
> 请问，小Cutey最多能采到多少果子?

Input:

> 第一行三个整数n,p,q，如题目描述中所述。
>
> 接下来n行，第i行一个整数Ai，表示第i棵树的情况，若Ai为-1，表示第i棵树上有毒蛇，否则Ai表示第i棵树上有Ai个果子。

Output:

> 若小Cutey能够到达第n棵树，则输出它最多能采到的果子数，否则输出-1。

Sample input:

```
5 1 2
1
-1
2
-1
1
```

Sample output:

```
4
```

Notes:

> 对于70%数据，n<=15；
>
> 对于100%数据，n<=50000，Ai<=100，A1>=0，An>=0，1<=p<=n，1<=q<=n。
>
> 本题时间限制1s，空间限制256MB。

Src:

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

#define MAX(a, b) ((a > b ? a : b))

int main()
{
    int n = 0, p = 0, q = 0;
    cin >> n >> p >> q;
    int *a = new int[n + 1]();
    int *max = new int[n + 1]();
    for (int i = 1; i <= n; i++)
    {
        cin >> a[i];
        if (a[i] == -1)
            max[i] = -1;
    }
    max[1] = a[1];
    for (int i = 2; i <= n; i++)
    {
        if (max[i] != -1)
        {
            if (i - p < 1 && i - q < 1)
                max[i] = -1;
            else if (i - p < 1 && i - q > 1)
            {
                if (max[i - q] == -1)
                    max[i] = -1;
                else
                    max[i] = max[i - q] + a[i];
            }
            else if (i - q < 1 && i - p > 1)
            {
                if (max[i - p] == -1)
```

```
                    max[i] = -1;
                else
                    max[i] = max[i - p] + a[i];
            }
            else
            {
                if (max[i - p] == -1 && max[i - q] == -1)
                    max[i] = -1;
                else
                    max[i] = MAX(max[i - p], max[i - q]) + a[i];
            }
        }
    }
    cout << max[n] << endl;
    return 0;
}
```

P1995.蛇形矩阵

Description:

Question:

> 一个漂亮的蛇形矩阵通常是长这样的：
>
>  P1995.蛇形矩阵
>
> 上图是一个7阶的蛇形矩阵。现在给定矩阵的阶数，你能给出蛇形矩阵长什么样吗？

Input:

> 一个正整数N,表示矩阵的阶数， 1<= N <= 20

Output:

> 对应阶数的蛇形矩阵

Sample input:

```
3
```

Sample output:

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

Notes:

> 无

Src:

```
1
```

P2147.旅行

Description:

Question:

> 你有m元钱，将要游览n个国家。每一个国家有一种商品，其中第i个国家商品的单价为ai元。每到一个国家，你会用手上的钱疯狂购买这个国家的商品，直到剩余的钱无法购买为止。
>
> 现在你要决定游览这n个国家的顺序，使得游览完n个国家后剩余的钱最多。

Input:

> 输入包含两行。
>
> 第一行两个整数n和m，表示国家数和钱数。
>
> 第二行n个整数分别为每个国家商品的单价。

Output:

> 一行一个整数，表示最多剩余多少元钱。

Sample input:

```
3 31
5 7 11
```

Sample output:

```
4
```

Notes:

> 1≤n≤1000,1≤m≤5000,1≤ai≤1000000000

Src:

```
1
```

P2317.Color

Description:

Question:

> Alice, Bob and Yazid are good friends.

> Each of them has a color, **r**ed, **g**reen or **b**lue. Everyone's color is different from the others'. They can describe their own colors in the format of "[name] is [color].", such as "Yazid is green.".
>
> Now they have made their descriptions in some order. After that, Yazid will do the following operations:
>
> \1. Connect the **3** sentences to get the **initial string**.
>
> \2. Remove all non-alphabetic characters.
>
> \3. Change all uppercase letters to lowercase.
>
> For example, if the initial string is "Yazid is green.Alice is red.Bob is blue.", then after Yazid's all operations, it will be turned to "yazidisgreenaliceisredbobisblue".
>
> Finally, Alice and Bob will insert any lowercase letters into any positions in this string to get the **final string**.
>
> You are given the final string. Your task is to find the initial string. In particular:
>
> \1. If there are multiple solutions, please output the minimum one in lexical order.
>
> \2. If there is no solution, please output "No solution." instead.

Input:

> The first line contains one integer T (T <= 500), representing the number of test cases. For each case:
>
> \1. One line of a string containing only lowercase letters, representing the final string.
>
> \2. It is guaranteed that the length of the final string won't exceed 600.

Output:

> For each case:
>
> \1. One line of a string, representing the initial string or `No solution.`.

Sample input:

```
4
aliceisredbobisblueyazidisgreen
aliceisgreenbobisgreenyazidisgreen
aliceisyellowbobisblueyazidisgreen
xxyazidxxisxxgreenxxbobisblueaxlxixcxexixsxrxexdx
```

Sample output:

```
Alice is red.Bob is blue.Yazid is green.
No solution.
No solution.
Yazid is green.Bob is blue.Alice is red.
```

Notes:

> 无

Src:

```
1
```

P2334.复杂的空调厂大赛

Description:

Question:

> 小Tyche在看书时，看见这么一道题：
>
> 夏日炎炎，空调机走俏。5家空调机厂的产品在一次质量评比活动中分获前5名。
>
> A厂的代表猜：E厂的产品稳获第1名；
>
> B厂的代表猜：我厂获得第2名；
>
> C厂的代表猜：A厂的质量最差；
>
> D厂的代表猜：C厂的产品不是最好的；
>
> E厂的代表猜：D厂会获第1名；
>
> 评比结果公布后发现，只有获第1名和第2名的两个厂的代表猜对了。
>
> 请编程给出A，B，C，D，E各是第几名？
>
> 小Tyche读完后，咔咔两下，把这题解出来了。他还觉得不过瘾，于是编了下面这道题：
>
> 夏日炎炎，空调机走俏。5家空调机厂的产品在一次质量评比活动中分获前5名。
>
> 在结果公布前，大家纷纷猜测每个人的名次，如下所示：
>
> X厂代表猜： Y厂获得第 Z 名。
>
> 其中，X和Y为"ABCDE"中的一个。每个厂都可以进行多次猜测。
>
> 评比结果公布后发现，只有获第1名和第2名的两个厂的代表猜对了至少一个，其他厂的所有猜测全部错了。
>
> 请编程给出A，B，C，D，E各是第几名？

> 那么，你能完成小Tyche新出的这道题么？

Input:

> 第一行输入一个整数N，表示接下来有N句话。 N <= 100;
>
> 接下来N行，每行包含三个元素 X（字符，为"ABCDE"中的一个）、 Y（字符，为"ABCDE"中的一个）、 Z（整数，1<=Z<=5）。元素之间用空格隔开。
>
> 注意：每个厂都可以猜测多次，每个厂对另外一个厂也可能进行多次猜测。

Output:

> 每组解输出一行，用五个数字，依次表示A~E的名次。即第一个数字表示A的名次，第二个数字表示B的名词，...，第五个数字表示E的名词。
>
> 此题可能有多解，多组解则按照字典序顺序，依次输出。每组解输出独立的一行。
>
> 输入保证存在解。

Sample input:

```
7
A A 1
B C 2
C B 3
D C 3
D C 2
D C 1
E E 5
```

Sample output:

```
15324
53124
```

Notes:

> 60分：忽略输入，直接做出小Tyche在书中看见的题目。
>
> 100分：按要求做出小Tyche出的新题。（p.s. 能做对小Tyche的新题，则书中的题必然也能对）

Src:

```
1
```

P2335.考试成绩可视化——垂直直方图

Description:

Question:

> 老师用同学们的程序，自动根据考试成绩，绘制考试分数直方图，真是酷毙了。只不过，老师更习惯看竖着的直方图，而之前程序中，直方图都是横着画的。那么，能不能绘制一个垂直的直方图呢？

所谓垂直的直方图，就是以并排的、宽度相等、高度与数值相关的矩形，来表现数列的图形。所有长方形的底端对齐，根据数值的不同，呈现不同的高度。（可以观察样例输入和输出，得到更加直观的了解）

为了简化问题，现在直方图的数据已经统计好了，你只需要直接画出垂直的直方图就可以。

Input:

第一行一个整数N。

第二行有N个整数a_i，表示直方图每列的高度。

0 < N <= 10, 0 <= a_i <= **10**

Output:

输出垂直的直方图。其中’’表示空白，’@’表示非空白。直方图的高度，为直方图中最高列的高度+1。

Sample input:

```
10
1 0 0 1 2 3 5 7 10 3
```

Sample output:

```
..........
........@.
........@.
........@.
.......@@.
.......@@.
......@@@.
......@@@.
.....@@@@@
....@@@@@@
@..@@@@@@@
```

Notes:

【样例解释】

首先，根据输入，我们发现，最大的数为10（第9个数），根据题目要求，直方图的高度为最高列的高度+1，所以，此时直方图的高度为11。

然后绘制直方图。输入中，第一列是1，所以第一列的高度是1。垂直直方图是从下往上延伸的，而又因为此时直方图高度为11，所以，第一列上面10个为空，输出为’’，最下面一个为‘@’，即

.

.

.

.

.

.

.

.

.

.

@

第2列和第3列都为0，所以全部为'.'

第9列，其高度为10，所以仅最上面的位置为'.'，其余10个位置为'@'，即

.

@

@

@

@

@

@

@

@

@

@

其余各列同理。将各列的图像，按顺序横向拼接，即形成最终的垂直直方图。

【数据规模】

50%的数据: N == 1

70%的数据: N <= 2

100%的数据: 0 < N <= 10

Src:

```
1
```

P2336.十进制的基数排序

Description:

Question:

> 基数排序（Radix sort）是一种经典的排序算法，下面用一个例子，讲解一下以10为基的基数排序的原理。

设待排序的数组为[62,14,59,88,16]，并假定有10个编号分别为0~9的桶，用来存放数字。基数排序的过程如下：

首先，依次处理所有数，根据每个数个位（即最右边第1位）上的数值,把它放到编号与个位数数值相等的桶中。完成这一步后，变成下面这样：

桶编号|__0__|__1__|__2__|__3__|__4__|__5__|__6__|__7__|__8__|__9__|

桶内容|_____|_____|__62_|_____|__14_|_____|__16_|_____|__88_|__59_|

然后，将各个桶中的数字，按照桶的编号顺序，依次取出来，得到下面的排序结果：

[62, 14, 16, 88, 59]

以上两步为基数排序的一趟排序过程。

接下来，进行第二趟排序。此时，我们只看[62, 14, 16, 88, 59]中这些数的十位数，和第一趟一样，将数组[62, 14, 16, 88, 59]中每一个数字都分配到对应的桶中，得到以下结果：

桶编号|__0__|__1__|__2__|__3__|__4__|__5__|__6__|__7__|__8__|__9__|

桶内容|_____|14, 16|__62_|_____|_____|__59_|__62_|_____|__88_|_____|

注意：14和16都分配到了编号为1的桶中。此时，先入桶的数（14）排在前面，后入桶的数（16）排在后面。具体来说，由于此趟排序，数组为[62, 14, 16, 88, 59]，按从左到右的顺序，14在前先进入，16在后后进入。因此该桶中，两个数的顺序为[14,16]。

然后，按照桶编号的顺序，把所有数依次取出来，得到下面这个结果：

[14, 16, 59, 62, 88]

至此，第二趟排序完成。

由于本示例中，所有数字都是两位数，所以经过两趟排序后，基数排序就完成了。

总结一下，对一个数组进行基数排序，需要进行多趟排序操作。

l 第一趟，根据数字最低位（个位数）排序

l 第二趟，根据数字次低位（十位数）排序

l 第三趟，根据数字第三低位（百位数）排序

l …

l 第N趟，根据数字第N低位排序

即从数字的低位到高位的顺序进行。排序进行的趟数，与数组中最大数的位数相同。

每趟排序分为两步：

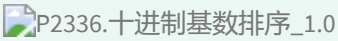
\1. 入桶：将数组中的数字，按从左到右的顺序，依次放入桶中。当一个桶中存放了多个数字时，先入桶的排在前面，后入桶的排在后面。

\2. 出桶：所有数字全部放入桶中之后，按照桶编号的从小到大顺序，依次将每个数字取出，得到本趟排序的结果。

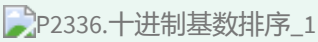
下面有几个演示动画，可以帮助理解算法：

算法演示1

初始状态：

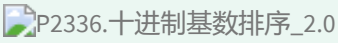


动画演示：

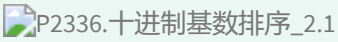


算法演示2

初始状态：

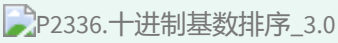


动画演示：

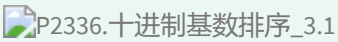


算法演示3

初始状态：



动画演示：



下面，请你完成基数排序算法，并将每一趟的中间排序结果，和最终排序结果输出。

Input:

第一行一个整数N表示有N个数。(N<=1000000)

第二行有N个整数a_i，表示数组中元素的值。(0 <= a_i <= 10000000)

Output:

> 输出多行，每行输出每一趟后的结果。数字之间用空格隔开。

Sample input:

```
10
278 109 63 930 589 184 505 269 8 83
```

Sample output:

```
930 63 83 184 505 278 8 109 589 269
505 8 109 930 63 269 278 83 184 589
8 63 83 109 184 269 278 505 589 930
```

Notes:

> 50分：N<=2000
>
> 70分：N<=100000
>
> 100分：N<=1000000

Src:

```
1
```

P2337.伪素数

Description:

Question:

> 给定M个数a1,a2,...,aM，这M个数互不相等。小明想知道1~N中有多少个数不能被这M个数整除。

Input:

> 第一行两个整数N, M。(N<=10000000, M<=10000)
>
> 接下来M行每行一个整数ai。 (2<=ai<=N)

Output:

> 一个整数，表示答案

Sample input:

```
20 4
6
11
3
7
```

Sample output:

```
11
```

Notes:

> 【样例解释】
>
> 1~20中，不被3、6、7、11整除的数，有1、2、4、5、8、10、13、16、17、19、20，共11个数。
>
> 【数据范围】
>
> 对于40%的数据 N<=10000, M<=100
>
> 对于60%的数据 N<=1000000, M<=1000
>
> 对于100%的数据 N<=10000000, M<=10000
>
> 提示：部分数据，只有使用筛法，才能在规定时间内得到结果。

Src:

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

int main()
{
    int n = 0, m = 0;
    cin >> n >> m;
    int num = n;
    bool *able = new bool[n + 1]();
    int *a = new int[m]();
    for (int i = 0; i < m; i++)
        cin >> a[i];
    for (int i = 0; i < m; i++)
        for (int j = 1; a[i] * j <= n; j++)
            if (!able[a[i] * j])
            {
                able[a[i] * j] = true;
                num--;
            }
    cout << num << endl;
    return 0;
}
```

P2338.算式计数

Description:

Question:

> 用1~2这2个数字，以及+、*两种运算符（共4种元素），可以构造出一些合法算式。一个合法算式的定义是：
>
> 1）运算符不相邻，如"1++2"不是合法算式；
>
> 2）开头不是运算符，如“+112”不是合法算式；

3）结尾不是运算符，如“122+”不是合法算式；

特别的，我们规定，单独一个数是合法算式，如“1212”是合法算式。

一些长度为5的合法算式的例子：“12 *21*”，“*1+2* 1”。

现在请你求出，长度为N的合法算式的个数。其中，所有元素都可重复使用。

由于在N较大时，合法算式数会较大，因此，我们要求你将合法算式数对1000007求余，输出求余后的结果。

Input:

一个整数N，表示算式的长度。

Output:

一个整数，表示长度为N的合法算式个数对1000007求余后的结果。

Sample input:

```
3
```

Sample output:

```
16
```

Notes:

【样例解释】

长度为3的算式，有两种类型：（以d表示数字，以x表示运算符）

‘ddd’型：如111、222等。此类算式个数为 2 * 2 * 2 = 8个。

‘dxd’型：如1+1、1*2等。此类算数个数为 2 * 2 * 2 = 8个。

因此，总合法算数个数为 8 + 8 = 16个。

由于 16 % 1000007 = 16，所以输出为16。

【数据范围】

对于60%的数据 N <= 35

对于70%的数据 N <= 2000

对于100%的数据 N <= 10000000

【注意】

由于int表示的数字范围是有限的（约10^9），当数字过大时，计算机程序在运行时会产生溢出错误（即超出计算机整数能表达的范围，在内存中的实际数值是错误的），从而导致答案错误。

所以，你需要在计算过程中，边计算边求余，只保留每步中间结果的余数，防止溢出。要特别注意运算过程中的乘法操作，它是最容易在计算中产生溢出的操作。

关于余数和求余操作，有以下性质：

l (a+b) % c == ((a%c) + (b%c))%c

l (a-b) % c == ((a%c) - (b%c) + c)%c （当 a >= b 时）

l (a*b) % c == ((a%c) * (b%c))%c

其中，a、b、c均为正整数。

此外，可以使用long long类型，代替int类型。long long类型也是整数，其比int能表示更大的数据范围（约10^18数量级），减少溢出风险。

【提示】

对于60%的数据使用long long类型代替int类型，只需要在最后让答案值%1000007。

本题使用递推思想来设计算法，会比较容易。

Src:

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

int main()
{
    int n = 0;
    cin >> n;
    int *num = new int[n + 1]();
    num[1] = 2;
    num[2] = 4;
    for (int i = 3; i <= n; i++)
        num[i] = ((2 * num[i - 1]) % 1000007 + 4 * num[i - 2] % 1000007) % 1000007;
    cout << num[n] << endl;
    return 0;
}
```

P2371.孪生素数对

Description:

Question:

> 差为2的两个素数（又称质数）被称为孪生素数对，例如3和5，11和13。
>
> 编写一个程序，从键盘输入两个正整数min和max（1≤min<max≤10^6），求解区间[min,max]内孪生素数对的个数。

Input:

> 两个正整数min和max（1≤min<max≤10^6）

Output:

> 输出一个整数，表示区间[min,max]内孪生素数对的个数

Sample input:

```
2 15
```

Sample output:

```
3
```

Notes:

> 注：在样例输出中，3表示在区间[2,15]内有三个孪生素数对，分别为3和5，5和7，11和13。而7和11则不是孪生素数对，因为它们之间相差为4。

Src:

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

int main()
{
    int l = 0, r = 0, num = 0;
    cin >> l >> r;
    bool *notPrime = new bool[r - l + 1]();
    for (int i = 2; i < sqrt(r); i++)
        for (int j = l / i; i * j <= r; j++)
            if (i * j >= l && i * j <= r && j != 1)
                notPrime[i * j - l] = true;
    int last = 0, first = 0;
    for (int i = 0; i <= r - l; i++)
        if (!notPrime[i])
        {
            last = i;
            first = i;
            break;
        }
    for (int i = first + 1; i <= r - l; i++)
        if (!notPrime[i])
```

```
        {
            if (i - last == 2)
                num++;
            last = i;
        }
    cout << num << endl;
    return 0;
}
```

P2372.预约讨论

Description:

Question:

> 小明参加了系里的学术新星计划，最近他有一个不成熟的idea，并打算找他的导师进行讨论。
>
> 现在是2019.1.1的0:00，在Google Calendar上小明看到他的导师有N个时间段已被占用（被占用的时间段之间互不重叠），以及导师将会在Y年M月D日H时m分出差。小明想知道，在导师出差前，他能预约到的最长讨论时间段为多少分钟？
>
> 注意：需要考虑跨日、跨月的情况。为了简便，**所有的测试用例的时间均在2019年内**。

Input:

> 第一行六个整数，依次为N, Y, M, D, H, m，分别表示被占用时间段数目N和导师出差的时间（Y年M月D日H时m分）。其中H是24小时制，m是60分钟制。
>
> 接下来N行，每行十个整数, $s1_{s5}$, $_{t1}t5$，表示s1年s2月s3日s4时s5分到t1年t2月t3日t4时t5分导师在忙。（所有被占用的时间段按时间的先后有序输入，不考虑无序的情况）

Output:

> 小明能预约到的最长讨论时间段的分钟数。

Sample input:

```
1 2019 1 2 0 0
2019 1 1 3 0 2019 1 1 17 0
```

Sample output:

```
420
```

Notes:

> 无

Src:

```

```

P2373.环形网络

Description:

Question:

> 本题首先需要输入一批操作命令来构建一个环形网络，例如图1展示了执行样例输入各操作命令后环形网络的构建情况，然后输入一个指定网络节点的标识，请问从网络接入点到指定的网络节点中间所经过的网络节点列表，以及总的网络传输延迟？
>
> 一个环形网络包含多个网络节点，相邻的网络节点之间只能按顺时针方向进行单向的信号传输。每个网络节点具有两个属性，一是每个节点具有唯一的“**网络标识**”，由6个小写字母组成；二是信号经过每个节点时，会有一定的“**传输延迟**”，由一个正整数表示。初始时环形网络为空。
>
> 对于一个环形网络，我们定义了如下5种操作命令：
>
> \1) **添加（**Append）：如果网络为空，添加一个节点并将其作为接入点；否则在当前接入点所在的网络节点**“逆时针**”方向添加一个相邻的节点，并保持接入点位置不变。如命令“Append 1 nameaa”表示添加一个传输延迟为1，网络标识为“nameaa”的节点。
>
> \2) **剔除（**Eliminate）**：以接入点所在的网络节点为1开始朝顺时针方向计数，每计到第m个节点就将其从当前网络中剔除，然后由下一个节点重新计数，直至剔除出n个网络节点为止（n小于网络中节点的数量）。若剔除的网络节点为接入点，则将顺时针方向的下一个网络节点设为接入点。如“Eliminate m n”表示剔除计数为m的n个网络节点。
>
> \3) **移动（**Move）**：将接入点朝顺时针或者逆时针方向移动r个节点，r是一个正整数。如“Move 0 3”表示将接入点按顺时针方向移动3个节点。其中第一个数字“0”表示移动方向，0表示顺时针方向（1表示逆时针方向）；第二个数字“3”表示移动的节点数目。
>
> \4) **排序（**Sort）**：将环形网络中网络节点按传输延迟由小到大排序，若传输延迟相同，则按节点网络标识的字典序排列。新的接入点仍指向排序前的网络节点。
>
> \5) **删除（**Delete）**：从环形网络中删除一个指定网络标识的节点。当指定网络标识的节点不存在时，则不执行任何操作；当删除的节点为接入点时，则将接入点顺时针方向第一个节点作为新的接入点；当删除网络中剩余的最后一个节点时，则把接入点置空。如“Delete namebb”表示删除环形网络中网络标识为“namebb”的网络节点。
>
> 注意：所有网络节点的网络标识都不相同，但网络传输延迟可以相同。

Input:

> 第一行一个整数T，表示命令数量。(1 <= T <= 1000)
>
> 接下来的T行，每行输入一个命令。
>
> 最后一行输入一个网络标识id，表示需要查询的网络节点。

Output:

> 输出一行或者多行。如果最终环形网络为空或指定查询网络标识对应的节点不在环形网络中，则输出“-1”；如果查询的节点就是接入点，则输出“0”；否则按序输出接入点到指定网络标识节点之间经过网络节点的网络标识列表（每行输出一条），以及总的传输延迟。

Sample input:

```
8
Append 1 nameaa
Append 3 bbbbbb
Append 5 aabbcc
Append 4 abcdef
Sort
Eliminate 2 1
Move 0 1
Delete nameaa
aabbcc
```

Sample output:

```
abcdef
4
```

Notes:

> **对于50%的数据，没有sort操作**
>
> P2373.环形网络

Src:

```

```

P2374.辩论赛评委

Description:

Question:

> 清华和北大之间要举行一场辩论赛，为保证比赛的公平性，辩论赛的“评委会成员”由两个大学共同选出。首先，辩论赛主办方面向两校师生和社会各界广泛召集评委会成员候选人。然后，清华和北大将分别对每个候选人进行评分，评分范围为0-20，0表示反对，20表示支持。最后，主办方综合考虑两个大学对候选人的评分，确定最后的评委会成员。
>
> 具体方法为：有n个候选人，每个候选人i有两个评分T_i和P_i，主办方从中选择m人组成评委会。设W是一种评委会方案，则W是候选人集合{1,2,…,n}的一个具有m个元素的子集。令T(W)=∑Ti，P(W)=∑Pi ,i∈W, 分别代表清华和北大给出m个人的总分。对于最佳的评委会W，绝对值|T(W)-P(W)|必须最小。如有多种评委会方案都满足绝对值|T(W)-P(W)|都最小，则应选择其中T(W)+P(W)最大的一种方案，若其中T(W)+P(W)最大的方案也包含多种，则选择含最小编号的一种方案。
>
> 主办方希望你编写程序根据每个候选人的评分确定评委会成员。

Input:

> 输入第一行两个整数n和m，n为候选人的总人数，m为评委会成员的人数。
>
> 接下来的n行，每行包含两个0-20之间的整数，分别为清华和北大对1到n号候选人的评分。
>
> 数据范围：1≤n≤200 , 1≤m≤20 , m≤n

Output:

> 输出包括两行，第一行为以空格分隔的两个整数T(W)和P(W)，分别表示清华和北大对选出的评委会成员的总分，第二行m个整数，同样以空格分隔，是按升序排列的评委会成员编号。

Sample input:

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

Sample output:

```
6 4
2 3
```

Notes:

> 无

Src:

```
1
```

P2375.压缩字符串

Description:

Question:

> 已知一个只包含大写字母 'A' - 'Z' 的字符串，现有一种压缩字符串的方法是：如果某个子串S连续出现了X次，就用 'X(S)' 来表示。例如AAAAAAAAAABABABCCD可以用10(A)2(BA)B2(C)D表示，则原字符串长度为18，压缩后的字符串长度为16。请注意在压缩后的字符串中，数字、字母和括号都需要算入压缩后的字符串长度。
>
> 对于一个字符串S，合法的压缩表示可能有很多种。例如AAAAAAAAAABABABCCD还可以表示成9(A)3(AB)CCD，则原字符串长度为18，压缩后的字符串长度为12。
>
> 此外，压缩表示方法也可以是嵌套的，例如HELLOHELLOWORLDHELLOHELLOWORLD可以表示成2(2(HELLO)WORLD)，则原字符串长度为30，压缩后的字符串长度为16。
>
> 如今我们需要知道对于给定的字符串，其最短表示方法的长度是多少?

Input:

> 第一行输入一个正整数T，表示需要测试的字符串数量。(1≤T≤10)
>
> 接下来的T行，每一行一个字符串S，长度不超过100。

Output:

> 对于每个测试字符串输出表示方法的最短长度（每行输出一个整数）。

Sample input:

```
3
AAA
AAAAAAAAAABABABCCD
HELLOHELLOWORLDHELLOHELLOWORLD
```

Sample output:

```
3
12
16
```

Notes:

> 注：在不压缩时，AAA的字符串长度为3，是最短表示。若将其压缩为3(A)，则压缩后的字符串长度为4，不是最短表示。

Src:

```
1
```

P1033.逻辑题

Description:

Question:

> 这题是一个圣诞礼物
>
> 啊不是。。。
>
> 这个题是一个逻辑题。
>
> 题目中的所有小写字母都代表一个公式。其中f~z可能被用以下方式定义：
>
> 1、非运算
>
> 字母1:!字母2
>
> 2、且运算
>
> 字母1:字母2&字母3
>
> 3、或运算
>
> 字母1:字母2|字母3
>
> 例如：
>
> f:a&b
>
> g:!f
>
> h:c&d
>
> z:g|h
>
> 那么公式z为

(!(a&b))|(c&d)

现在保证后面出现的公式中只包含a_e以及前面的公式。问a~e中的哪些为真时能够z为真。

Input:

第一行一个正整数n，表示定义的数目。接下来n行，每行描述一个定义。格式见题目描述。保证z被定义。保证每个字母最多被定义一次。

Output:

输出若干行，每行若干个a~e的字母，表示一个答案。表示以这些公式为真时z为真。需要输出所有的答案。

答案按以下顺序排列：一个答案中的不同的字母，按字母顺序排列，中间不插入空格。对于每个答案，不含a的排在含a的前面；如果情况相同则比较b，不含b的排在含b的前面。依此类推。

Sample input:

```
4
f:a&b
g:!f
h:c&d
z:g|h
```

Sample output:

```

e
d
de
c
ce
cd
cde
b
be
bd
bde
bc
bce
bcd
bcde
a
ae
ad
ade
ac
ace
acd
acde
abcd
abcde
```

Notes:

【样例说明】

注意输出的第一行是一个空行，即a~e都为假时z为真。由于原式中没有提到e，因此不管e是否为真都不会影响到结果。

Src:

```
1
```

P1099.机器指令

Description:

Question:

A同学在一台简陋的计算机上运行程序H。（同《计算机科学导论》一书的附录C）假设这台机器的CPU仅有16个通用寄存器（register）（用十六进制的0到F编号），主存只有256个内存单元（memory cell）（用十六进制00到FF编号），每个机器指令长两个字节（用十六进制数表示），各种指令的解释如下：

1RXY　将内存单元XY里的数据复制到寄存器R中

2RXY　将数据XY存至寄存器R中

3RXY　将寄存器R中的数据复制到内存单元XY中

40RS　将寄存器R中的数据复制到寄存器S中

5RST　将寄存器S和寄存器T中储存的整数相加并存放在寄存器R中

BRXY　如果寄存器R中的内容和寄存器0中的内容相同，则程序计数器指向XY内存单元（执行JUMP操作），否则程序按原来流程继续运行

C000　结束程序

在一个指令循环中，CPU每次提取程序计数器指向的内存单元和下一个内存单元中两个十六进制数组成的四位十六进制数作为机器指令并执行，若在指令执行时未进行JUMP操作，则把程序计数器向后移两个单元。一旦程序开始运行，CPU将进入指令循环直到出现“结束程序”的命令为止（C000）。现在这个程序H只需使用从00到0F的内存单元，且在程序开始前程序计数器（program counter）已经指向了00单元。输入程序H运行前00到0F内存单元里存放的数据，请输出程序H结束后00到0F内存单元里的数据。

Input:

输入一行16个十六进制数（每个十六进制数占两位，可能以0开头），用空格隔开，分别表示程序H运行前00到0F内存单元中存放的数据。

Output:

输出一行16个十六进制数（每个十六进制数占两位，可能以0开头），用空格隔开，分别表示程序H结束后00到0F内存单元中存放的数据。

Sample input:

```
1   2A 99 3A 00 1A 03 3A 02 C0 00 00 00 00 00 00 00
```

Sample output:

```
99 99 00 00 1A 03 3A 02 C0 00 00 00 00 00 00 00
```

Notes:

> 【样例解释】
>
> 运行指令2A99：将十六进制数99存至寄存器A中；
>
> 运行指令3A00：将寄存器A中的数据（99）复制到内存单元00中；
>
> 运行指令1A03：将内存单元03里的数据（00）复制到寄存器A中；
>
> 运行指令3A02：将寄存器A中的数据（00）复制到内存单元02中；
>
> 运行指令C000：结束程序。

Src:

```
1
```

Final_2020

P2428.【20'】过冬

Description:

Question:

> 土地上开了n朵名贵的花，但随着冬天的来临难以抵挡住寒风，勤劳的你需要给它们修建一个大棚让他们平安度过冬天。现在给你这n朵花的坐标，请你设计一个最小的**正方形**大棚，使得所有花能够平安度过冬天。（在正方形边上的花认为在大棚内部）
>
> **（正方形平行于坐标轴，不用考虑旋转）**

Input:

> 第一行是一个整数n，表示有n朵花。
>
> 接下来有n行，每行两个整数x,y，表示花的坐标。

Output:

> 输出一个整数，最小的**正方形**大棚的边长。

Sample input:

```
5
-1 1
2 3
4 2
2 2
-4 0
```

Sample output:

```
8
```

Notes:

> 80%的数据满足1<=n<=103, -109 <= x, y <= 109
>
> 100%的数据满足 1<= n <= 105, -109 <= x, y <= 109

Src:

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

#define MAX(a, b) ((a > b) ? a : b)

int main()
{
    int n = 0, xmax = 0, ymax = 0, xmin = 0, ymin = 0, x = 0, y = 0;
    cin >> n >> x >> y;
    xmax = x;
    xmin = x;
    ymax = y;
    ymin = y;
    for (int i = 1; i < n; i++)
    {
        cin >> x >> y;
        if (x > xmax)
            xmax = x;
        if (x < xmin)
            xmin = x;
        if (y > ymax)
            ymax = y;
        if (y < ymin)
            ymin = y;
    }
    int len = MAX(xmax - xmin, ymax - ymin);
    cout << len << endl;
    return 0;
}
```

P2429.【20'】统计区间字母个数

Description:

Question:

> 设有一个字符串S，它全部由小写字母组成。
>
> 现在，老师想知道，字符串S从下标L到下标R的区间中，不同的小写字母各出现了多少次。（注意：字符串下标从0开始）
>
> 老师一共会向你询问Q次，每次都会告诉你区间的起始下标L和终止下标R。请你回答每次询问。

Input:

> 第一行一个字符串S（长度 <= 1000）
>
> 第二行为一个整数Q（Q <= 100）
>
> 接下来有Q行，每行两个整数L和R。（0 <= L, R < 字符串S的长度）

Output:

> 对于每次询问，输出一行结果，表示每种小写字母出现的次数。
>
> 输出格式为“<小写字母>:<出现次数>”（用英文冒号分隔字母和次数，中间没有空格），请按从a到z的次序，输出每种小写字母，以及对应出现的次数。
>
> 每两种小写字母的结果之间用一个空格隔开。
>
> 忽略出现次数为0的小写字母。

Sample input:

```
abcdefgaaa
2
0 0
1 8
```

Sample output:

```
a:1
a:2 b:1 c:1 d:1 e:1 f:1 g:1
```

Notes:

> 第一次询问，区间为[0, 0]，即"[a]bcdefgaaa"，因此只出现一个小写字母a。
>
> 第二次询问，区间为[1, 8]，即"a[bcdefgaa]a"，因此出现a两次，b~g各一次。

Src:

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

int main()
{
    char str[1010] = {0};
    int q = 0, num[26] = {0}, l = 0, r = 0;
    cin >> str >> q;
    for (int i = 0; i < q; i++)
    {
        memset(num, 0, sizeof(num));
        cin >> l >> r;
        for (int j = l; j <= r; j++)
            num[str[j] - 'a']++;
        for (int j = 0; j < 26; j++)
            if (num[j])
                cout << (char)('a' + j) << ':' << num[j] << ' ';
        cout << endl;
    }
```

```
21        return 0;
22    }
```

P2430.【20'】素数间距

Description:

Question:

> “素数（也称质数）”是指在大于1的自然数中，除了1和它本身以外没有其他因数的自然数。如果两个素数之间没有其他素数，则称这两个素数为一对“**相邻素数**”。例如，2和3是一对“相邻素数”，3和7则不是“相邻素数”，因为在3和7之间有素数5。
>
> 请你编写一个程序，对于给定两个数字L和U所限定的区间，你需要：
>
> （1）找到一对“相邻素数”C1和C2（L <= C1 < C2 <=U），使得**C2-C1最小**。如果在这个区间中有多对“相邻素数”都是间距最小的，则选择其中**C1+C2最小**的一对。
>
> （2）找到一对“相邻素数”D1和D2（L <= D1 < D2 <= U），使得**D2-D1最大**。如果在这个区间中有多对“相邻素数”都是间距最大的，则选择其中**D1+D2最小**的一对。
>
> 注意：在指定区间内可能没有“相邻素数”，此时请按题目要求输出。

Input:

> 输入两个正整数L和U，且L<U。

Output:

> 输出包括两行，第一行为区间内间距最小的“相邻素数”（数值小的在前，数值大的在后，中间为空格），第二行为区间内间距最大的“相邻素数”（数值小的在前，数值大的在后，中间为空格）。
>
> 若区间内没有相邻素数，则仅输出-1。

Sample input:

```
1    2  17
```

Sample output:

```
1    2  3
2    7  11
```

Notes:

> 【样例二】
>
> 输入：
>
> 14 17
>
> 输出：
>
> **-1**

【数据范围】

50%的数据满足：1 ≤ L < U ≤ 105；

80%的数据满足：1 ≤ L < U ≤ 106；

100%的数据满足：1 ≤ L < U ≤ 109，且(U-L) ≤ 106。

Src:

```cpp
//有一个点没过
#include <iostream>
#include <cmath>
using namespace std;

bool IsPrime(int n)
{
    for (int i = 2; i <= sqrt(n); i++)
        if (n % i == 0)
            return false;
    return true;
}

int main()
{
    int l = 0, u = 0;
    int min[2] = {0}, max[2] = {0}, prime[10000] = {0};
    int num = 0;
    cin >> l >> u;
    for (int i = l; i <= u; i++)
    {
        if (IsPrime(i))
        {
            prime[num++] = i;
            if (num != 1)
            {
                if (((prime[num - 1] - prime[num - 2]) < (min[1] - min[0])) ||
min[1] == 0)
                {
                    min[0] = prime[num - 2];
                    min[1] = prime[num - 1];
                }
                if ((prime[num - 1] - prime[num - 2]) > (max[1] - max[0]))
                {
                    max[0] = prime[num - 2];
                    max[1] = prime[num - 1];
                }
            }
        }
    }
    if (max[1] == 0)
        cout << -1 << endl;
    else
    {
        cout << min[0] << ' ' << min[1] << endl;
        cout << max[0] << ' ' << max[1] << endl;
    }
```

```
47        return 0;
48    }
```

P2431.【25'】矩阵查询

Description:

Question:

> 给你一个n*m的矩阵A=(a_{ij})，矩阵左上角坐标为(1,1)，右下角坐标为(n,m)。现在有三种对矩阵的操作：
>
> 1. $A\ x_1\ y_1\ x_2\ y_2\ d$，表示在左上角坐标为(x1,y1)右下角坐标为(x2,y2)的举行中所有数加d，即，对于所有$(x_1 \le i \le x_2, y_1 \le j \le y_2)$，执行$a_{ij} = a_{ij} + d$；
> 2. $E\ x_1\ x_2$，表示将第x_1行元素与第x_2行元素交换；
> 3. $Q\ x\ y$，表示询问矩阵中坐标为(x,y)处的值为多少，即a_{xy}的值。
>
> 例如，给定矩阵
>
> $$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$
>
> 对于操作“$A\ 1\ 1\ 2\ 2\ 1$”，矩阵A变为
>
> $$A = \begin{bmatrix} 2 & 3 & 3 \\ 5 & 6 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$
>
> 接着对于操作“$E\ 1\ 3$”，矩阵A变为
>
> $$A = \begin{bmatrix} 7 & 8 & 9 \\ 5 & 6 & 6 \\ 2 & 3 & 3 \end{bmatrix}$$
>
> 对于操作“$Q\ 3\ 3$”，输出3。

Input:

> 第一行两个整数n，m，表示矩阵有n行m列。
>
> 接下来的n行，每行m个数，表示矩阵里元素的初始值。
>
> 第n+2行有一个整数q，表示操作的次数。
>
> 接下来的q行是对操作的描述。

Output:

> 对于每个询问输出结果。

Sample input:

```
3 4
1 2 3 4
5 6 7 8
9 10 11 12
5
A 1 1 1 2 -1
E 2 3
A 3 3 3 3 2
Q 2 2
Q 3 1
```

Sample output:

```
10
5
```

Notes:

> 50%的数据满足： 1 ≤ n,m ≤ 10, |a_ij| ≤ 200000 , q ≤ 10；
>
> 100%的数据满足：1 ≤ n,m ≤ 100, |a_ij| ≤ 200000 , q ≤ 100, |d| ≤ 1000。

Src:

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

int n = 0, m = 0, q = 0;

void swap(int &a, int &b)
{
    int tmp = a;
    a = b;
    b = tmp;
}

void A(int **a, int x1, int y1, int x2, int y2, int d)
{
    for (int i = x1; i <= x2; i++)
        for (int j = y1; j <= y2; j++)
            a[i][j] += d;
}

void E(int **a, int x1, int x2)
{
    for (int i = 1; i <= m; i++)
        swap(a[x1][i], a[x2][i]);
}

void Q(int **a, int x, int y)
{
    cout << a[x][y] << endl;
}

int main()
{
```

```
    cin >> n >> m;
    int **a = new int *[n + 1]();
    for (int i = 1; i <= n; i++)
        a[i] = new int[m + 1]();
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            cin >> a[i][j];
    cin >> q;
    for (int i = 0; i < q; i++)
    {
        char cmd = '\0';
        cin >> cmd;
        if (cmd == 'A')
        {
            int x1 = 0, x2 = 0, y1 = 0, y2 = 0, d = 0;
            cin >> x1 >> y1 >> x2 >> y2 >> d;
            A(a, x1, y1, x2, y2, d);
        }
        else if (cmd == 'E')
        {
            int x1 = 0, x2 = 0;
            cin >> x1 >> x2;
            E(a, x1, x2);
        }
        else if (cmd == 'Q')
        {
            int x = 0, y = 0;
            cin >> x >> y;
            Q(a, x, y);
        }
    }
    return 0;
}
```

P2432.【15'】擀面皮

Description:

Question:

> 有一块1x1的方形面团（不考虑面团的厚度），其口感值为0。擀面师傅要将其擀成一个N x M（纵向长N，横向宽M）的面皮。师傅的擀面手法娴熟，每次下手，要么横向擀一下（使得横向长度增加1），要么纵向擀一下（使得纵向长度增加1）。此外，当面团（皮）的大小为a x b时，往横向擀一下会使得面的口感值上升H_ab，而往纵向擀一下则会使口感值上升V_ab。
>
> 现在，请你来将1x1的面团擀成N x M面皮。显然，从1x1的面团擀成N x M的面皮有多种不同的操作序列可以实现，不同操作序列下得到的最终面皮口感值也可能是不同的。请问最终得到的N x M面皮，口感值最高可为多少？

Input:

> 第一行两个整数N，M，表示要擀出来面皮的大小（纵向长N，横向宽M）。
>
> 接下来有N行，每行M个数。第a行第b列的数值H_ab，表示当面皮大小为a x b时，横向擀一下后，面皮口感的上升值。

再接下来有N行，每行M个数。第a行第b列的数值V_ab，表示当面皮大小为a x b时，纵向擀一下后，面皮口感的上升值。

（0 < N, M < 1000，0 <= H_ab, V_ab <= 1000）

Output:

输出最终得到的N x M面皮的最高的口感值。

Sample input:

```
2 3
1 2 3
4 5 6
11 12 13
14 15 16
```

Sample output:

```
20
```

Notes:

【样例解释】

一共三种擀面方法：

纵横横：11+4+5=20

横纵横：1+12+5=18

横横纵：1+2+13=16

【样例二】

输入：

3 3

1 0 2

2 0 2

2 2 0

0 2 2

1 2 1

2 1 2

输出：

7

【样例二解释】

> 最优擀面方法为:横(1) + 纵(2) + 纵(2) + 横(2) = 7

Src:

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

#define MAX(a, b) ((a > b) ? a : b)

int n = 0, m = 0, **h = NULL, **v = NULL, **val = NULL;

int main()
{
    cin >> n >> m;
    h = new int *[n + 1]();
    v = new int *[n + 1]();
    val = new int *[n + 1]();
    for (int i = 1; i <= n; i++)
    {
        h[i] = new int[m + 1]();
        v[i] = new int[m + 1]();
        val[i] = new int[m + 1]();
    }
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            cin >> h[i][j];
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            cin >> v[i][j];
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
        {
            if (i == 1 && j == 1)
                continue;
            if (i == 1)
                val[i][j] = val[i][j - 1] + h[i][j - 1];
            else if (j == 1)
                val[i][j] = val[i - 1][j] + v[i - 1][j];
            else
                val[i][j] = MAX(val[i - 1][j] + v[i - 1][j], val[i][j - 1] + h[i][j - 1]);
        }
    cout << val[n][m] << endl;
    return 0;
}

```

Final_2021

P2435.【20'】圆内点的数量统计

Description:

Question:

>  P2435.圆内点.0

Sample input:

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

Sample output:

```
3 2 2
```

Notes:

> 样例1示例图：
>
>  P2435.圆内点.1
>
> 样例2：
>
> 输入：
>
> 5
>
> 1 1
>
> 2 2
>
> 3 3
>
> 4 4
>
> 5 5
>
> 4
>
> 1 2 2
>
> 2 2 2
>
> 4 3 2
>
> 4 3 3
>
> 输出：
>
> 2 3 2 4

> 示例图：
>  P2435.圆内点.2

Src:

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

int dis_2(int x1, int y1, int x2, int y2)
{
    return ((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
}

int main()
{
    int n = 0, m = 0;
    cin >> n;
    int **dot = new int *[n + 1]();
    for (int i = 1; i <= n; i++)
        dot[i] = new int[2]();
    for (int i = 1; i <= n; i++)
        cin >> dot[i][0] >> dot[i][1];
    cin >> m;
    int **circle = new int *[m + 1]();
    for (int i = 1; i <= m; i++)
        circle[i] = new int[4]();
    for (int i = 1; i <= m; i++)
        cin >> circle[i][0] >> circle[i][1] >> circle[i][2];
    for (int i = 1; i <= m; i++)
        for (int j = 1; j <= n; j++)
            if (dis_2(circle[i][0], circle[i][1], dot[j][0], dot[j][1]) <=
circle[i][2] * circle[i][2])
                circle[i][3]++;
    for (int i = 1; i <= m; i++)
        cout << circle[i][3] << ' ';
    return 0;
}
```

P2436.【20'】天数统计

Description:

Question:

>  P2436.天数统计

Sample input:

```
20220412 20220414
```

Sample output:

```
3
```

Notes:

样例二：

输入：

```
20220101 20250101
```

输出：

1097

解释：

2022、2023为平年共$365 \times 2 = 730$天，2024年为闰年366天，再加上2025年的1月1日，总天数为$730 + 366 + 1 = 1907$。

Src:

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

const int month_day[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

bool IsLeap(int year)
{
    if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0))
        return true;
    return false;
}

int convert(int n)
{
    int day = n % 100;
    n = n / 100;
    int month = n % 100;
    n = n / 100;
    int year = n;
    int sum = 0;
    for (int i = 2022; i <= year; i++)
    {
        sum += 365;
        if (IsLeap(i - 1))
            sum += 1;
    }
    for (int i = 1; i < month; i++)
        sum += month_day[i];
    sum += day;
    return sum;
}

int main()
{
    int day1 = 0, day2 = 0;
    cin >> day1 >> day2;
    cout << convert(day2) - convert(day1) + 1 << endl;
    return 0;
}
```

P2437.【25'】单词查找

Description:

Question:

>  P2437.单词查找.0

Sample input:

```
8
qprogram
mpdthkjy
awrdghji
rrbosfgz
ghgrgwth
ozzzzrzo
rwdfrgag
pyyygggm
```

Sample output:

```
*program
mp******
a*r*****
r**o****
g***g***
o*****r**
r******a*
p******m
```

Notes:

> 样例中的三个“program”出现位置如下：
>
>  P2437.单词查找.1

Src:

```
//写的有点复杂，可以再简练些
#include <iostream>
using namespace std;

const char text[] = "program";

void row(char **matrix, bool **IsText, int n, int x)
{
    for (int i = 1; i <= n; i++)
    {
        if (matrix[x][i] == text[0] && i + 6 <= n)
        {
            int p = 1, q = i + 1;
            for (; p <= 6; p++, q++)
                if (matrix[x][q] != text[p])
                    break;
            if (p == 7)
            {
```

```
                for (int k = 0; k <= 6; k++, i++)
                    IsText[x][i] = true;
            }
        }
    }
    for (int i = n; i >= 1; i--)
    {
        if (matrix[x][i] == text[0] && i - 6 >= 1)
        {
            int p = 1, q = i - 1;
            for (; p <= 6; p++, q--)
                if (matrix[x][q] != text[p])
                    break;
            if (p == 7)
            {
                for (int k = 0; k <= 6; k++, i--)
                    IsText[x][i] = true;
            }
        }
    }
}

void col(char **matrix, bool **IsText, int n, int x)
{
    for (int i = 1; i <= n; i++)
    {
        if (matrix[i][x] == text[0] && i + 6 <= n)
        {
            int p = 1, q = i + 1;
            for (; p <= 6; p++, q++)
                if (matrix[q][x] != text[p])
                    break;
            if (p == 7)
            {
                for (int k = 0; k <= 6; k++, i++)
                    IsText[i][x] = true;
            }
        }
    }
    for (int i = n; i >= 1; i--)
    {
        if (matrix[i][x] == text[0] && i - 6 >= 1)
        {
            int p = 1, q = i - 1;
            for (; p <= 6; p++, q--)
                if (matrix[q][x] != text[p])
                    break;
            if (p == 7)
            {
                for (int k = 0; k <= 6; k++, i--)
                    IsText[i][x] = true;
            }
        }
    }
}

void slope_pos(char **matrix, bool **IsText, int n)
{
```

```
    int b = 0;
    for (b = 1 - n; b <= n - 1; b++)
    {
        for (int y = (b <= 0 ? 1 - b : 1); y + b <= n && y <= n; y++)
        {
            if (matrix[y + b][y] == text[0] && y + b + 6 <= n && y + 6 <= n)
            {
                int p = 1, q = y + 1;
                for (; p <= 6; p++, q++)
                    if (matrix[q + b][q] != text[p])
                        break;
                if (p == 7)
                {
                    for (int k = 0; k <= 6; k++, y++)
                    {
                        IsText[y + b][y] = true;
                    }
                }
            }
        }
    }
    for (b = 1 - n; b <= n - 1; b++)
    {
        for (int y = (b > 0 ? n - b : n); y + b >= 1 && y >= 1; y--)
        {
            if (matrix[y + b][y] == text[0] && y + b - 6 >= 1 && y - 6 >= 1)
            {
                int p = 1, q = y - 1;
                for (; p <= 6; p++, q--)
                    if (matrix[q + b][q] != text[p])
                        break;
                if (p == 7)
                {
                    for (int k = 0; k <= 6; k++, y--)
                    {
                        IsText[y + b][y] = true;
                    }
                }
            }
        }
    }
}

void slope_neg(char **matrix, bool **IsText, int n)
{
    int b = 0;
    for (b = 2; b <= 2 * n; b++)
    {
        for (int y = (b < n + 1 ? 1 : b - n); - y + b <= n && y <= n && y >= 1
&& -y + b >= 1; y++)
        {
            if (matrix[-y + b][y] == text[0] && -y + b - 6 <= n && y + 6 <= n &&
-y + b - 6 >= 1 && y + 6 >= 1)
            {
                int p = 1, q = y + 1;
                for (; p <= 6; p++, q++)
                    if (matrix[-q + b][q] != text[p])
                        break;
```

```
                if (p == 7)
                {
                    for (int k = 0; k <= 6; k++, y++)
                    {
                        IsText[-y + b][y] = true;
                    }
                }
            }
        }
    }
    for (b = 2; b <= 2 * n; b++)
    {
        for (int y = (b < n + 1 ? b - 1 : n); - y + b <= n && y <= n && y >= 1
&& -y + b >= 1; y--)
        {
            if (matrix[-y + b][y] == text[0] && -y + b + 6 <= n && y - 6 <= n &&
-y + b + 6 >= 1 && y - 6 >= 1)
            {
                int p = 1, q = y - 1;
                for (; p <= 6; p++, q--)
                    if (matrix[-q + b][q] != text[p])
                        break;
                if (p == 7)
                {
                    for (int k = 0; k <= 6; k++, y--)
                    {
                        IsText[-y + b][y] = true;
                    }
                }
            }
        }
    }
}

void change(char **matrix, bool **IsText, int n)
{
    for (int i = 1; i <= n; i++)
    {
        row(matrix, IsText, n, i);
        col(matrix, IsText, n, i);
    }
    slope_pos(matrix, IsText, n);
    slope_neg(matrix, IsText, n);
}

int main()
{
    int n = 0;
    cin >> n;
    char **matrix = new char *[n + 1]();
    bool **IsText = new bool *[n + 1]();
    for (int i = 1; i <= n; i++)
    {
        matrix[i] = new char[n + 1]();
        IsText[i] = new bool[n + 1]();
    }
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
```

```
            cin >> matrix[i][j];
    change(matrix, IsText, n);
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
            cout << (IsText[i][j] ? matrix[i][j] : '*');
        cout << endl;
    }
    return 0;
}
```

P2438.【25'】小明的橡皮泥

Description:

Question:

> 

Sample input:

```
5
CREATE RED DEER
DUP 1 YELLOW
CREATE YELLOW TIGER
CRASH 2
ORDER
```

Sample output:

```
RED DEER
YELLOW TIGER
```

Notes:

> 样例解释：
>
> 

Src:

```
//可以再简练些
#include <iostream>
#include <cstring>
using namespace std;

struct node
{
    char color[25];
    char shape[25];
    node *next;
};

node *head = NULL;
int num = 0;
```

```
15
16  void swap(node *p1, node *p2)
17  {
18      if ((p1 == head || p2 == head) && (p1->next == p2 || p2->next == p1))
19      {
20          node *p = NULL, *q = NULL;
21          if (p1 == head)
22          {
23              p = p1;
24              q = p2;
25          }
26          else
27          {
28              p = p2;
29              q = p1;
30          }
31          node *k = q->next;
32          head = q;
33          q->next = p;
34          p->next = k;
35      }
36      else if (p1 == head || p2 == head)
37      {
38          if (p1 == head)
39          {
40              node *k1 = p1->next, *k2 = p2->next;
41              node *q2 = head;
42              while (1)
43              {
44                  if (q2->next == p2)
45                      break;
46                  else
47                      q2 = q2->next;
48              }
49              q2->next = p1;
50              p1->next = k2;
51              p2->next = k1;
52              head = p2;
53          }
54          else if (p2 == head)
55          {
56              node *k1 = p1->next, *k2 = p2->next;
57              node *q1 = head;
58              while (1)
59              {
60                  if (q1->next == p1)
61                      break;
62                  else
63                      q1 = q1->next;
64              }
65              q1->next = p2;
66              p2->next = k1;
67              p1->next = k2;
68              head = p1;
69          }
70      }
71      else if (p1->next == p2 || p2->next == p1)
72      {
```

```
            if (p1->next == p2)
            {
                node *q = head, *k = p2->next;
                while (q->next != p1)
                    q = q->next;
                q->next = p2;
                p1->next = k;
                p2->next = p1;
            }
            else if (p2->next == p1)
            {
                node *q = head, *k = p1->next;
                while (q->next != p2)
                    q = q->next;
                q->next = p1;
                p2->next = k;
                p1->next = p2;
            }
        }
        else
        {
            node *k1 = p1->next, *k2 = p2->next;
            node *q1 = head, *q2 = head;
            while (1)
            {
                if (q1->next == p1 && q2->next == p2)
                    break;
                if (q1->next != p1)
                    q1 = q1->next;
                if (q2->next != p2)
                    q2 = q2->next;
            }
            q1->next = p2;
            p2->next = k1;
            q2->next = p1;
            p1->next = k2;
        }
    }

    void create(char c[], char s[])
    {
        node *n = new node();
        memset(n->color, 0, sizeof(n->color));
        memset(n->shape, 0, sizeof(n->shape));
        n->next = NULL;
        strcpy(n->color, c);
        strcpy(n->shape, s);
        if (!head)
            head = n;
        else
        {
            node *p = head;
            for (int i = 1; i < num; i++)
                p = p->next;
            p->next = n;
        }
        num++;
    }
```

```
131
132 int main()
133 {
134     int n = 0;
135     cin >> n;
136     char cmd[25] = {0};
137     for (int i = 1; i <= n; i++)
138     {
139         memset(cmd, 0, sizeof(cmd));
140         cin >> cmd;
141         if (!strcmp(cmd, "CREATE"))
142         {
143             char color[25] = {0}, shape[25] = {0};
144             cin >> color >> shape;
145             create(color, shape);
146         }
147         else if (!strcmp(cmd, "DUP"))
148         {
149             int m = 0;
150             char color[25] = {0};
151             cin >> m >> color;
152             node *p = head;
153             for (int i = 1; i < m; i++)
154             {
155                 if (!p)
156                     break;
157                 p = p->next;
158             }
159             create(color, p->shape);
160         }
161         else if (!strcmp(cmd, "CRASH"))
162         {
163             int m = 0;
164             cin >> m;
165             node *p = head;
166             if (m == 1)
167             {
168                 if (head)
169                 {
170                     head = p->next;
171                     delete[] p;
172                 }
173             }
174             else
175             {
176                 for (int i = 1; i < m - 1; i++)
177                     p = p->next;
178                 node *q = p->next;
179                 p->next = q->next;
180                 delete[] q;
181             }
182             num--;
183         }
184         else if (!strcmp(cmd, "ORDER"))
185         {
186             node *p = head;
187             for (int i = 1; i < num; i++)
188             {
```

```
                    node *q = p->next;
                    for (int j = i; j < num; j++)
                    {
                        if (!strcmp(p->color, q->color))
                        {
                            for (int k = 0;; k++)
                            {
                                if (p->shape[k] < q->shape[k])
                                    break;
                                if (p->shape[k] > q->shape[k])
                                {
                                    swap(p, q);
                                    node *tmp = p;
                                    p = q;
                                    q = tmp;
                                    break;
                                }
                            }
                        }
                        else
                        {
                            for (int k = 0;; k++)
                            {
                                if (p->color[k] < q->color[k])
                                    break;
                                if (p->color[k] > q->color[k])
                                {
                                    swap(p, q);
                                    node *tmp = p;
                                    p = q;
                                    q = tmp;
                                    break;
                                }
                            }
                        }
                        q = q->next;
                    }
                    p = p->next;
                }
            }
        }
        node *p = head;
        while (p)
        {
            cout << p->color << ' ' << p->shape << endl;
            p = p->next;
        }
        return 0;
    }
```

P2439.【10'】小明的周末

Description:

Question:

> P2439.小明的周末

Sample input:

```
5 2
6 1 8 2 1
```

Sample output:

```
1 4
```

Notes:

> **样例**1解释：
>
> 根据“参加**编号连续**的k项活动”要求，可供小明选择的活动安排有：
>
> 方案一、周六参加编号1和2的k项活动，时长分别为6和1；周日参加编号3和4的k项活动，时长分别为8和2。
>
> 方案二、周六参加编号1和2的k项活动，时长分别为6和1；周日参加编号4和5的k项活动，时长分别为2和1。
>
> 方案三、周六参加编号2和3的k项活动，时长分别为1和8；周日参加编号4和5的k项活动，时长分别为2和1。
>
> 方案四、周六参加编号3和4的k项活动，时长分别为8和2；周日参加编号1和2的k项活动，时长分别为6和1。
>
> 方案五、周六参加编号4和5的k项活动，时长分别为2和1；周日参加编号1和2的k项活动，时长分别为6和1。
>
> 方案六、周六参加编号4和5的k项活动，时长分别为2和1；周日参加编号2和3的k项活动，时长分别为1和8。
>
> 在这6种活动安排方案中，总时间最少的是方案二和方案五，根据题目要求，应选择周六起始活动编号较小的方案二，输出周六和周日的起始活动编号1和4。
>
> **样例**2输入：
>
> 5 2
>
> 1 1 1 1 1
>
> **样例**2输出：
>
> 1 3
>
> **样例**2解释：
>
> 任意两个连续的活动所花费的时间均为2，周六周日加起来的总时间均为4，因此只需要选择周六与周日活动编号均较小的方案，即周六选择编号为1和2的活动，周日选择编号为3和4的活动。结果输出周六和周日的起始活动编号1和3。

Src:

```cpp
//应当使用dp，考场上看出来了没写出来。使用暴力破解有四个点时间超限
#include <iostream>
#include <limits.h>
using namespace std;

int main()
{
    int n = 0, k = 0;
    cin >> n >> k;
    int *time = new int[n + 1]();
    int min = INT_MAX;
    int min1 = 0, min2 = 0;
    for (int i = 1; i <= n; i++)
        cin >> time[i];
    for (int i = 1; i <= n; i++)
    {
        int sum1 = 0;
        if ((i - 1 < k && n - i - k + 1 < k) || (i + k - 1 > n))
            continue;
        for (int j = i; j <= i + k - 1; j++)
            sum1 += time[j];
        for (int j = 1; j <= n; j++)
        {
            int sum2 = sum1;
            if ((j >= i && j <= i + k - 1) || (j + k - 1 >= i && j + k - 1 <= i + k - 1) || (j + k - 1 > n))
                continue;
            for (int t = j; t <= j + k - 1; t++)
                sum2 += time[t];
            if (sum2 < min)
            {
                min1 = i;
                min2 = j;
                min = sum2;
            }
        }
    }
    cout << min1 << ' ' << min2 << endl;
    return 0;
}
```