一、编程题(共 5 题，共 100 分)

1. 最近的斐波那契数

最近的斐波那契数

斐波那契数列 F_n 的定义为：对 $n \geq 0$ 有 $F_{n+2} = F_{n+1} + F_n$，初始值为 $F_0 = 0$ 和 $F_1 = 1$。所谓与给定的整

本题就请你为任意给定的整数 N 找出与之最近的斐波那契数。

时间限制：1000
内存限制：65536

输入
输入在一行中给出一个正整数 N（$\leq 10^8$）。

输出
在一行输出与 N 最近的斐波那契数。如果解不唯一，输出最小的那个数。

样例输入

```
305
```

样例输出

```
233
```

提示
样例解释 部分斐波那契数列为 { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ... }。可见 23

试题编号：20241207-3-01

试题类型：编程题

标准答案：

试题难度：一般

试题解析：

展示地址：点击浏览

考生答案：（此题已作答）

考生得分：20

是否评分：已评分

```
#include<iostream>
#include<bits/stdc++.h>
#include<math.h>
using namespace std;
int main(){
    int n[101];
    n[0]=0,n[1]=1;
    for(int i=2;i<43;i++){
        n[i]=n[i-1]+n[i-2];

    }
    int N;
    cin>>N;
    int min_diff=9999999;
    int temp[101]={0},j=0;
    for(int i=0;i<43;i++){
        if(abs(n[i]-N)<min_diff){
            min_diff=abs(n[i]-N);
            temp[j++]=n[i];
        }
    }
    cout<<temp[j-1];
    return 0;
}
```

评价描述：

2. 构造性证明

构造性证明

关于数学定理证明，也有高下之分。最暴力的证明方法是“构造性证明”，即当需要证明某种解存在时，直

下面有一个定理：

设 ai（i=1, … , 5）均为正实数。则一定存在 4 个互不相同的下标 i、j、k、l，使得 |ai / aj - ak / al| < 1/2

作为程序员，就请你编写程序构造出正确的下标，验证这个结论。

时间限制：1000
内存限制：65536

输入
输入在一行中顺序给出 5 个正实数。为保证计算中不产生浮点溢出，我们令输入的数字在 [10^{-10}, 10^{10}] [

输出
在一行中首先输出使得定理结论成立的下标有多少套，随后输出最小的一套下标。数字间以 1 个空格分隔
$i_k < j_k$。

样例输入

```
3.12 5.27 0.0007 9825.4413 10
```

样例输出

```
18 1 4 3 2
```

提示
样例解释： 易验证 $|a_1/a_4-a_3/a_2|=|3.12/9825.4413 - 0.0007/5.27| < 1/2$。满足条件的解有 18 个，例如

试题编号：20241207-3-02

试题类型：编程题

标准答案：

试题难度：一般

试题解析：

展示地址：点击浏览

考生答案：（此题已作答）

考生得分：20

是否评分：已评分

评价描述：

```
#include <iostream>
```

```cpp
#include <vector>
#include <algorithm>
#include <cmath>
#include <iomanip>
using namespace std;

const double EPS = 1e-9;


// 比较两个双精度浮点数是否相等（考虑精度误差）
bool doubleEqual(double a, double b) {
    return fabs(a - b) < EPS;
}

int main() {
    vector<double> nums(5);
    for (int i = 0; i < 5; ++i) {
        cin >> nums[i];
    }
    int count = 0;
    vector<int> result(4);
    // 四重循环遍历所有可能的四个下标组合
    for (int i = 0; i < 5; ++i) {
        for (int j = 0; j < 5; ++j) {
            if (i == j) continue;
            for (int k = 0; k < 5; ++k) {
                if (k == i || k == j) continue;
                for (int l = 0; l < 5; ++l) {
                    if (l == i || l == j || l == k) continue;
                    double val = fabs(nums[i] / nums[j] - nums[k] / nums[l]);
                    if (val < 0.5) {
                        count++;
                        if (count == 1) {
                            result = {i + 1, j + 1, k + 1, l + 1};
                        }
                    }
                }
            }
        }
    }
    cout << count << " ";
    for (int idx : result) {
        cout << idx << " ";
```

```
    }
    return 0;
}
```

3. 环形公路出口

环形公路出口

一条环形高速路上有 N 个出口。给定任意一对出口，请你算出这两个出口之间的最短距离。

时间限制：1000
内存限制：65536

输入
输入第一行给出区间 [3,10^5] 内的整数 N，以及 N 个整数距离 D_1 D_2 … D_N，其中 D_i 是第 i 和第 i+1 个出
给出一对出口的编号（出口从 1 到 N 顺序编号）。题目保证公路全长不超过 10^7。

输出
输出 M 行，每行给出对应输入的一对出口之间的最短距离。

样例输入

```
5 1 2 4 14 9
3
1 3
2 5
4 1
```

样例输出

```
3
10
7
```

试题编号：20241207-3-03

试题类型：编程题

标准答案：

试题难度：一般

试题解析：

展示地址：点击浏览

考生答案：（此题已作答）

考生得分：20

是否评分：已评分

评价描述：

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

int main() {
    int N;
    cin >> N;
    vector<int> distances(N);
    for (int i = 0; i < N; ++i) {
        cin >> distances[i];
    }
    vector<int> prefixSum(N + 1, 0);
    for (int i = 0; i < N; ++i) {
        prefixSum[i + 1] = prefixSum[i] + distances[i];
    }

    int M;
    cin >> M;
    for (int i = 0; i < M; ++i) {
        int start, end;
        cin >> start;
        cin >> end;
        if (start > end) {
            swap(start, end);
        }
        int sum = prefixSum[end - 1] - prefixSum[start - 1];
        int total = prefixSum.back();
        cout << min(sum, total - sum) << endl;
    }

    return 0;
```

}

4.

子串与子列

子串和子列

子串是一个字符串中连续的一部分，而**子列**是字符串中保持字符顺序的一个子集，可以连续也可以不连续

现给定一个字符串 S 和一个子列 P，本题就请你找到 S 中包含 P 的最短子串。若解不唯一，则输出起点

时间限制：1000
内存限制：65536

输入
输入在第一行中给出字符串 S，第二行给出 P。S 非空，由不超过 10^4 个小写英文字母组成；P 保证是 S

输出
在一行中输出 S 中包含 P 的最短子串。若解不唯一，则输出起点最靠左边的解。

样例输入

```
atpaaabpabttpcat
pat
```

样例输出

```
pabt
```

试题编号：20241207-3-04

试题类型：编程题

标准答案：

试题难度：一般

试题解析：

展示地址：点击浏览

考生答案：（此题已作答）

考生得分：0

是否评分：已评分

评价描述：

```
#include <iostream>
using namespace std;
int main() {
    int point, MIN = 10000, ansl = 0, ansr = 0;
    string S, P;
    cin >> S >> P;
    for (int i = 0; i <= S.size() - P.size(); i++) {
        if (S[i] == P[0]) {
            point = 1;
            for (int j = i + 1; j < S.size(); j++) {
                if (j - i== MIN) break;
                if (S[j] == P[point]) point++;
                if (point == P.size()) {
                    ansl = i, ansr = j;
                    MIN = j - i ;
                    break;
                }
            }
        }
    }
    for (int i = ansl; i <= ansr; i++) cout << S[i];
    return 0;
}
```

5. 拼大数

拼大数

如何随机生成一个有 n 位数的大数呢？一种方法是，找到 n 个小朋友，每人发一张卡片，卡片一面写着编号（这里假设小
张，按他们的编号升序排列，显示他们自己写的数字。

但是，让十万个孩子都按指令行动，可太难了。结果是卡片乱七八糟满墙都是，有些甚至显示的不是正确的面。例如第 2
的大数。

时间限制：6000
内存限制：65536

输入

输入第一行给出一个正个整数 n (≤ 10^5)，随后 n 行，每行按 n1 n2 的格式给出一张卡片两面的数字。

输出

在一行中输出我们真想要拼成的 n 位大数。 如果卡片两面都是 1 位数，那么就很难说哪个数字是编号，哪个数字是小朋友

样例输入

```
12
7 11
8 9
3 1
2 12
4 6
10 0
5 1
2 5
6 8
1 4
7 2
9 3
```

样例输出

```
359114268072
```

试题编号：20241207-3-05

试题类型：编程题

标准答案：

试题难度：一般

试题解析：

展示地址：点击浏览

考生答案：（此题已作答）

考生得分：0

是否评分：已评分

评价描述：