Cặp số đặc biệt

Hai số A và B được gọi là một cặp số đặc biệt nếu như chúng có cùng số chữ số và tích $C = A \times B$ bao gồm tất cả các chữ số của A và B được viết theo một thứ tự nào đó. Chẳng hạn, 15 và 93 là một cặp số đặc biệt có 2 chữ số vì $15 \times 93 = 1395$ là một số gồm 4 chữ số $1, 3, 5, 9$ của A và B.

Nhiệm vụ của bạn là liệt kê K cặp số đặc biệt có N chữ số bất kì.

Tên bài: SPECNUM

Input: Standard Input

- Một dòng duy nhất chứa hai số nguyên K và N.

Output: Standard Output

- In ra K dòng, mỗi dòng in ra ba số A, B, C cách nhau bởi dấu cách.

Giới hạn:

- $1 \leq K \leq 100$.
- $2 \leq N \leq 100$.

Chấm điểm:

- Output của bạn phải đảm bảo $C = A \times B$. Bạn sẽ bị 0 điểm cho test tương ứng nếu có một dòng mà $C \neq A \times B$.
- Output của bạn phải đảm bảo C có $2 \times N$ chữ số và bao gồm tất cả các chữ số của A và B. Bạn sẽ bị 0 điểm cho test tương ứng nếu có một dòng không như vậy.
- Output của bạn không được in ra hai cặp số A, B giống nhau trên hai dòng khác nhau, và không được in ra cặp giống nhau mà đổi chỗ A và B. Chẳng hạn, nếu bạn in ra $15\ 93\ 1395$ trên một dòng và $93\ 15\ 1395$ trên một dòng khác, bạn sẽ bị 0 điểm cho test tương ứng.

Sample Input	Sample Output
3 2	15 93 1395 27 81 2187 60 21 1260

Trò chơi trên bảng

Bạn được cho một bảng $N \times M$ gồm toàn 0 và 1. Bạn chơi một trò chơi trên bảng này, được chia thành các nước đi, nhiệm vụ là đưa tất cả các ô trên bảng về 0. Ở mỗi nước đi, bạn được quyền chọn một hình chữ nhật con trong bảng có kích thước $R \times C$ và đảo tất cả các ô trong hình chữ nhật con đó – 0 thành 1, 1 thành 0. Ở hai lượt khác nhau, bạn không được phép chọn hai hình chữ nhật có kích thước giống nhau. Nói cách khác, nếu bạn đã chọn hình chữ nhật có kích thước $R_1 \times C_1$ ở một lượt và hình chữ nhật có kích thước $R_2 \times C_2$ tại một lượt khác thì bắt buộc $R_1 \neq R_2$ hoặc $C_1 \neq C_2$.

Bạn hãy viết chương trình để chơi trò chơi này nhé.

Tên bài: BOARDGAME

Input: Standard Input

- Dòng đầu tiên chứa hai số nguyên N, M.
- N dòng tiếp theo, mỗi dòng chứa M số nguyên là 0 hoặc 1 mô tả bảng.

Output: Standard Output

- Dòng đầu tiên in ra số nguyên K, số lượng nước đi của bạn.
- K dòng tiếp theo, mỗi dòng in ra bốn số nguyên x, y, R, C cho biết tại nước đi tương ứng bạn chọn đảo số ở hình chữ nhật có kích thước $R \times C$ và có ô góc trái trên nằm ở hàng x cột y.

Giới hạn:

- $1 \leq N, M \leq 1000$.

Chấm điểm:

- Output của bạn phải đảm bảo hình chữ nhật bạn chọn tại mỗi nước đi nằm trọn vẹn trong bảng. Nếu hình chữ nhật của bạn nằm ngoài bảng, bạn sẽ bị 0 điểm cho test tương ứng.
- Output của bạn phải đảm bảo tất cả các cặp R, C phải khác nhau. Nếu bạn in ra R, C giống nhau trên hai dòng khác nhau, bạn sẽ bị 0 điểm cho test tương ứng.
- Bạn phải đảm bảo chơi như cách của bạn sẽ đưa tất cả các ô trên bảng về 0. Nếu tất cả các ô của bảng không về 0, bạn sẽ bị 0 điểm cho test tương ứng.

Standard Input	Standard Output
3 4 1 0 0 0 0 1 1 0 0 0 0 0	2 1 1 1 1 2 2 1 2

Giết Quicksort

Hẳn thầy Hoàng đã dạy bạn thuật toán Quicksort rồi chứ? Bạn cũng biết là thuật toán Quicksort có độ phức tạp $O(N \times logN)$ trong trường hợp trung bình, nhưng là $O(N^2)$ trong trường hợp xấu nhất. Vì vậy thầy Hoàng dạy là bạn phải chọn chốt (pivot) một cách ngẫu nhiên để không thể nào mà tìm test làm cho Quicksort có độ phức tạp $O(N^2)$ được. Tuy nhiên, điều bạn có thể chưa biết là chỉ chọn một cách ngẫu nhiên mà không dùng lệnh srand(time(NULL)) thì vẫn có thể tìm test để hack được. Nhiệm vụ của bạn trong bài này là giết một code Quicksort như vậy.

Bài toán là cho N số nguyên, hãy in ra N số đó theo thứ tự tăng dần. Một coder đã giải bài này bằng C++ như sau:

```
#include <bits/stdc++.h>
using namespace std;

const int MAX_N = 100000;
int n;
int a[MAX_N];
int timer;

void Quicksort(int* a, int l, int h) {
     if(l >= h) {
          return;
     }
     int pivotIndex = l + rand() % (h - l + 1);
     int pivot = a[pivotIndex];
     swap(a[l], a[pivotIndex]);
     int i = l;
     int j = h;
     while(i < j) {
          while(i < j && a[j] > pivot) {
               j--;
               timer++;
          }
          if(i < j) {
               a[i] = a[j];
               i++;
               timer++;
          }
          while(i < j && a[i] < pivot) {
               i++;
               timer++;
          }
          if(i < j) {
               a[j] = a[i];
               j--;
```

```
                timer++;
            }
        }
        a[i] = pivot;
        Quicksort(a, l, i - 1);
        Quicksort(a, i + 1, h);
}

int main() {
        cin >> n;
        for(int i = 0; i < n; i++) {
            cin >> a[i];
        }
        timer = 0;
        Quicksort(a, 0, n - 1);
        for(int i = 0; i < n; i++) {
            cout << a[i] << " ";
        }
        cout << endl;
}
```

Trong code trên, biến $timer$ sử dụng để đo thời gian chạy hàm Quicksort. Biến này sẽ tăng thêm 1 mỗi lần i và j thay đổi. Nhiệm vụ là bạn phải sinh test sao cho sau khi code trên chạy thì $timer > 5 \times 10^6$.

Tên bài: SORTKILL

Input: Standard Input

- Một số nguyên N duy nhất là độ dài của dãy số.

Output: Standard Output

- In ra N số nguyên tùy ý là dãy số của bạn. Bạn sẽ được điểm nếu như sau khi chạy code trên với dãy số của bạn, $timer > 5 \times 10^6$.

Giới hạn:

- $9 \times 10^4 \leq N \leq 10^5$.
- Số bạn in ra nằm trong khoảng $[1, 10^9]$.