一般来说，选择排序是每一次从待排序的数据元素中选出最小的一个元素，存放在序列的起始位置，直到全部待排序的数据元素排完。

那么核心代码就应该是类似于：

```
for (int i = 0; i < a.length - 1; i++) {
            minIndex = i; // 无序区的最小数据数组下标
            for (int j = i + 1; j < a.length; j++)
                // 在无序区中找到最小数据并保存其数组下标
                if (a[j] < a[minIndex])
                    minIndex = j;
            int t = a[i];
            a[i] = a[minIndex];
            a[minIndex] = t;
        }
```

然而，jxtxzzw喜欢把选择排序写成：

```
for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j)
            if (a[i] > a[j]) {
            int t = a[i];
            a[i] = a[j];
            a[j] = t;
            }
```

很显然，这样的写法也是正确的。

但是，jxtxzzw的这种写法，虽然写起来方便，但是效率会比较低。

请你自行分析效率低的原因，并验证。

输出格式

没有输入。

你需要构造一组测试数据，来证明jxtxzzw的选择排序效率比较差。

你需要输出 2 行。

第 1 行是一个正整数 n ，表示待排序的数字个数。

n 必须是正整数，且必须是 1⩽n⩽100000000 范围。

第 2 行是 n 个整数，表示 n 个待排序的数字。

可以是正整数、负整数或 0 ，但必须在 `int` 范围，数字与数字之间用空格分隔。

你构造的数据需要保证：

- 能够使一般的选择排序在 5 秒内得到正确的结果；
- 但是jxtxzzw的选择排序需要大于 5 秒的时间才能得到正确的结果。

例如，一种合法的输出是：

```
10
1 2 3 -1 3 2 1 -1 -1 -1
```

但这组数据就连jxtxzzw的选择排序也能够在 5 秒钟内得到结果。

注意你构造的排序规模不能太大，否则可能连一般的选择排序都没法在 5 秒钟之内得到结果，但也不能太小，否则连jxtxzzw的选择排序都可以在 5 秒钟之内得到结果。

你可能需要在巧妙地进行数据的分布和排列，而不是一味研究数据规模。