Problem 1. 去重

描述

现有一组从小到大排列的、可能有重复元素的数列。请编写程序，对数列去重，输出去重后有多少个元素。

输入格式

第 1 行，一个整数 N，代表数列有多少个元素；

第 2 行，N 个空格隔开的整数 V[0] ~ V[N-1]，分别为数列各个元素的值（确保了从小到大排列），值确保可以用 `int` 存储。

输出格式

一行，只有一个整数，即去重后数列还有多少个元素。

样例输入

```
11
1 5 5 6 8 9 9 9 9 9 9
```

样例输出

```
5
```

样例解释

去重后，数列元素从小到大依次为 `1 5 6 8 9`，共 5 个元素，因此输出 5。

数据范围和限制

测试点 1

数据范围：N = 1

时间限制：1s 内

内存限制：128MB 内

测试点 2 ~ 测试点 8

数据范围：N > 1，上限未知，但确保 N 一定能被 `int` 存储，且确保存储 N 个元素时不会超出内存限制

时间限制：1s 内

内存限制：128MB 内

测试点 9 ~ 测试点 10

数据范围：N >= 1，上限未知，但确保 N 一定能被 `int` 存储

时间限制：1s 内

内存限制：1MB 内

Problem 2. 多边形

描述

以下给出了几个类的函数接口定义和功能描述，请根据描述实现这些类。这些类用于进行二维空间上几何图形的计算。注意，你需要自行定义成员变量，并且函数的声明中可能需要添加 `const`。

1. 类 `point`

代表一个二维空间上的坐标 (x, y)。本题中，坐标系 (0, 0) 处是左上角，向右、向下为正方向。

- 构造函数 `point()`：构造坐标 (0, 0)
- 构造函数 `point(double x, double y)`：根据参数构造坐标 (x, y)
- 成员函数 `double get_x()`：返回 x
- 成员函数 `double get_y()`：返回 y
- 成员函数 `void set_x(double x)`：设定 x
- 成员函数 `void set_y(double y)`：设定 y

2. 抽象类 `polygen`

代表一个凸多边形。

- 构造函数 `polygen(std::vector<point> points)`：根据一组顺时针方向的凸多边形顶点坐标构造多边形，顶点的数量一定大于等于 3（该接口假设输入数据一定是顺时针方向且是凸多边形）
- 成员函数 `int sides()`：返回该多边形边的个数
- 成员函数 `double area()`：计算该多边形面积，请将此函数定义为纯虚函数，它将由子类实现

3. 类 `rectangle`

继承自 `polygen`，代表一个长方形。

- 构造函数 `rectangle(point left_top, point right_bottom)`：由左上角坐标和右下角坐标构造长方形
- 覆盖成员函数 `double area()`：计算该长方形面积

4. 类 `triangle`

继承自 `polygen`，代表一个三角形。

- 构造函数 `triangle(point p1, point p2, point p3)`：接收顺时针方向三个坐标，由这三个坐标构造三角形

- 覆盖成员函数 `double area()`：基于海伦公式（见提示）计算三角形面积

递交说明

你递交的代码将被作为一个头文件，被我们的测试代码 include、编译并测试。因此代码中应当包含且仅包含这些类的声明和实现，不要包含 `int main()` 等其他代码。

测试点说明

测试点 1：正确实现了 `point` 类

测试点 2：正确实现了 `point` 类且支持 `const`

测试点 3：在测试点 1 基础上，正确实现了 `polygen` 和 `rectangle` 类

测试点 4：在测试点 1 基础上，正确实现了 `polygen` 和 `triangle` 类

测试点 5：正确实现了所有类

测试点 6：正确实现了所有类且支持 `const`

提示

海伦公式

假设有一个三角形，边长分别为 a, b, c，三角形的面积 A 可由以下公式求得：

$$A = \sqrt{s(s-a)(s-b)(s-c)}，其中\ s = \frac{a+b+c}{2}$$

Problem 3. Naive 数据库

描述

助教在写一个简易数据库，它可以存储 N 行 M 列任意长度字符串数据（数据确保只有字母、数字或下划线），每一列都有一个名字。给定列名和值（值确保只有字母、数字或下划线），该数据库软件可以瞬间查询到符合条件（即该行该列的值和查询值完全一致）的那一行（确保至多只有一行会匹配，但可能没有一行能匹配）。现在助教向该数据库发起了 P 次查询请求，每次请求查询的都是查固定的某一列（确保查询的列名一定和数据中某个列的名称一致），请编写代码输出查询结果。

输入格式

第 1 行，三个空格隔开的整数 N、M、P，分别为数据的行数、数据的列数、查询数量；

第 2 行：M 个空格隔开的字符串，分别是数据中每一列的名称；

第 3 行：一个字符串，代表需要查询的那列的名称；

接下来 i 行 (0 <= i < N) 行：每行 M 个空格隔开的字符串，分别是数据第 i 行里各列的值；

接下来 j 行 (0 <= j < P) 行：每行代表一次查询，每行只有一个字符串，即待查询的值（查询的列在第 3 行已给出）。

输出格式

输出共 P 行，每行对应各次查询结果。对于每个查询，若该列中不存在这个值，则结果为 `Not Found`（注意大小写）；若存在这个值，则结果为匹配到的那一行各列的值（按输入顺序原样输出、空格分隔）。

输入样例

```
6 4 5
Id Name Homework_Score Exam_Score
Name
66123 Sweet 100 100
000 swx 99 100
2333 luban_7hao 60 cheat
666 a_ke 70 absent
12321 zhang_fei 0 80
31461200 da_qiao 80 90
sweet
Sweet
xiao_qiao
da_qiao
666
```

输出样例

```
Not Found
66123 Sweet 100 100
Not Found
31461200 da_qiao 80 90
Not Found
```

样例解释

第一次查询 Name 为 `sweet` 的行，未找到（大小写必须完全一致），因此输出 `Not Found`。

第二次查询 Name 为 `Sweet` 的行，与数据第 1 行 `66123 Sweet 100 100` 匹配，因此输出该行。

第三次查询 Name 为 `xiao_qiao` 的行，未找到（必须完全一致），因此输出 `Not Found`。

第四次查询 Name 为 `da_qiao` 的行，与数据第 6 行 `31461200 da_qiao 80 90` 匹配，因此输出该行。

第五次查询 Name 为 `666` 的行，未找到（`Name` 列下没有值为 `666` 的行），因此输出 `Not Found`。

数据范围和限制

测试点 1 ~ 测试点 3

数据范围：0 < N, M, P <= 1000，所有字符串长度都小于 100

时间限制：1s 内

内存限制：128MB 内

测试点 4 ~ 测试点 6

数据范围：N, M > 0，0 < P <= 1000，确保存储数据时不会超出内存限制

时间限制：1s 内

内存限制：128MB 内

测试点 7 ~ 测试点 10

数据范围：N, M > 0，0 < P <= 1000000，确保存储数据时不会超出内存限制

时间限制：1s 内

内存限制：128MB 内