活动须知

1. 用 `p1.cpp`、`p2.cpp`、……、`p6.cpp` 命名你的程序。超过 100KB 的代码不予评测。
2. 程序直接使用 cin 或 scanf 读入、cout 或 printf 输出，不要打开输入或输出文件。
3. 所有测试点运行时间限制为 1s (以实际评测机为准)，内存限制为 1GB。
4. 随题册下发有额外的样例数据。

第一题 (p1.cpp, 75 分): 幸运数字

如果一个 n 位正整数恰好由数字 $0, 1, 2, \ldots, n-1$ 组成，Dr. X 就称它为“幸运数字”，例如：

- 1023 是一个幸运数字，因为它是一个四位数，且恰好由数字 0, 1, 2, 3 组成。
- 123 不是幸运数字，因为三位幸运数字应该由数字 0, 1, 2 组成。
- 012 不是幸运数字，因为我们只考虑不含前导零的正整数。

现在，给定两个正整数 a 和 b，请你计算 $a, a+1, a+2, \ldots, b$ 中幸运数字的数量。

输入格式

输入两个空格分隔的正整数 a 和 b。

输出格式

输出一个整数，表示 $a, a+1, a+2, \ldots, b$ 中幸运数字的数量。

样例输入 1

```
4 202
```

样例输出 1

```
4
```

- 在 4 和 202 之间，幸运数字有 $10, 102, 120, 201$。

样例输入 2

```
1 100000
```

样例输出 2

```
119
```

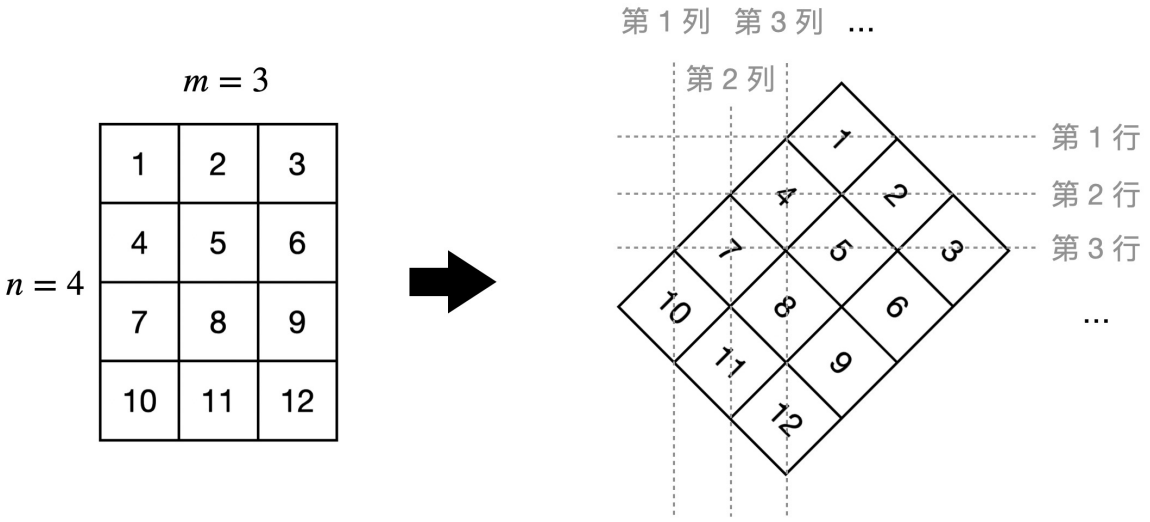
数据规模

- 对于 100% 的数据，满足 $1 \le a \le b \le 1,000,000$。

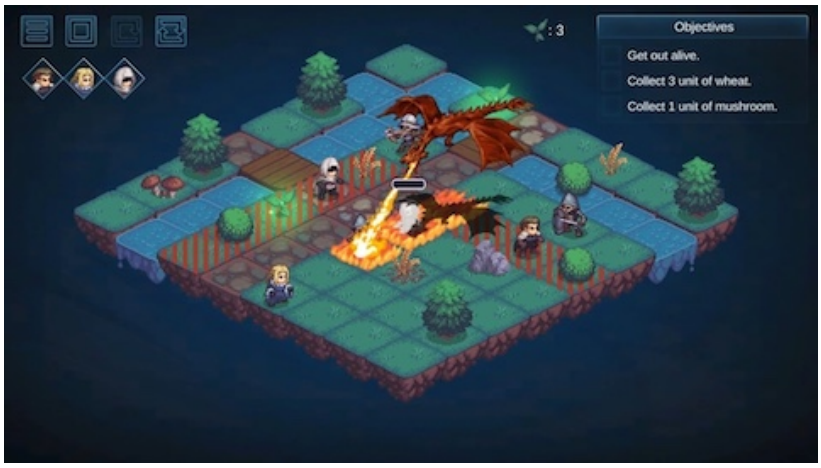
第二题 (p2.cpp, 75 分): 坐标变换

今天我们玩的许多游戏都是真正的 3D 游戏。在计算机能高效处理三维对象之前，人们就发明了斜 45 度视角的 “2.5D” 游戏，不需要额外的 3D 加速硬件也能流畅运行。

2.5D 游戏的地图是一个 n 行、m 列的网格。我们首先将网格里的格子按照从上到下、从左到右的顺序从 1 开始编号，再把网格顺时针 (向右) 旋转 45 度，就得到了游戏的地图。此时，网格的行和列发生了变化：旋转后最顶部的格子位于第一行；最左侧的格子位于第一列，如下图所示：



Dr. X 希望实现自己的 2.5D 游戏引擎，他希望你计算 n 行 m 列网格中，编号为 k 的格子，在旋转 45 度后的新网格中的行和列。



输入格式

输入空格分隔的整数 n, m 和 k，分别表示网格的行数、列数和格子的编号。

输出格式

输出空格分隔的整数 x, y，表示编号为 k 的格子在旋转 45 度后的新网格中的坐标。

样例输入 1

```
4 3 7
```

样例输出 1

```
3 2
```

样例输入 1

```
16 16 233
```

样例输出 1

```
23 10
```

数据规模

- 对于 60% 的数据，$n, m \leq 100$。
- 对于 100% 的数据，$1 \leq n, m \leq 32,768$, $1 \leq k \leq n \times m$。

第三题 (p3.cpp, 75 分): 美味水果

Dr. X 收到了一份礼物：n 个水果，其中第 i 个水果的好吃程度为 x_i。新鲜的水果会随时间变得不如最初好吃：

- 每天，Dr. X 可以选择吃掉一个水果，并记录下该天吃掉的水果的好吃程度。
- 没有被吃掉的每个水果，好吃程度将在第二天变为 $y = \lfloor \sqrt{x} \rfloor$，即“开根号取整”：$y$ 是满足 $y^2 \leq x$ 的最大整数。

请计算，在所有可能的吃水果顺序中，Dr. X 最多能获得多少好吃程度的总和。

输入格式

输入包含两行：第一行，一个整数 n，表示水果数量；第二行，n 个用空格分隔的整数 $x_1, x_2, ..., x_n$，表示每个水果的初始好吃程度。

输出格式

输出一个整数，表示能够获得的好吃程度总和的最大值。

样例输入 1

```
2
100 10
```

样例输出 1

```
103
```

- 在第一天，Dr.X 吃掉第一个水果，好吃程度为 100，另一个水果在第二天吃，好吃程度为 $\lfloor \sqrt{10} \rfloor = 3$，吃完所有水果，好吃程度的总和为 103。

样例输入 2

```
6
1 3 7 10 15 21
```

样例输出 2

```
28
```

数据规模

- 对于 40% 的数据，$1 \le n \le 100$。
- 对于 100% 的数据，$1 \le n \le 100,000$，水果的好吃程度 $1 \le x_i \le 1,000,000,000$。

提示

```cpp
// #include <cmath>
// 或 #include <bits/stdc++.h>
int y = (int)sqrt(x);
```

可以计算 $y = \lfloor \sqrt{x} \rfloor$。

第四题 (p4.cpp, 75 分): 成语接龙

小朋友们都很喜欢成语接龙，但如果遇到可以无限接龙的“防不胜防”、“忍无可忍”和“为所欲为”，游戏就会陷入僵局。Dr. X 发明了改进版的成语接龙：把 16 个汉字写在 4 行 4 列的方阵中，要求：

- 第一行 (从左往右读) 是一个四字成语。
- 最后一列 (从上往下读) 是一个四字成语。
- 最后一行 (从右往左读) 是一个四字成语。
- 第一列 (从下往上读) 是一个四字成语。
- 对角线 (从左上到右下读) 是一个四字成语。
- 对角线 (从右上到左下读) 是一个四字成语。
- 上面 6 个成语各不相同。

下面是一个满足条件的成语接龙：

Dr. X 找到了一本成语词典，你能帮 Dr.X 编程找到词典中所有满足条件的成语接龙方阵吗?

输入格式

输入第一行是整数 n，表示成语词典中成语的数量。

接下来 n 行，每行四个空格分开的、长度不超过 8 的字符串 (由小写字母和数字组成)。相同的字符串代表同一个汉字。

输出格式

输出一个整数，代表不同成语接龙方阵的数量。对于两个 4×4 的方阵，只要存在任意位置的汉字不同，就认为是不同的方阵。

样例输入 1

```
10
hua xiang niao yu
yu miao tian xia
xia li ba ren
ren mian tao hua
hua qian yue xia
yu chu jing ren
ji xiang ru yi
xin xiang shi cheng
cong ming ling li
jian kang cheng zhang
```

样例输出

```
1
```

样例输入 2

(输入为随题册下发的 idioms.txt)

样例输出 2

```
58
```

数据规模

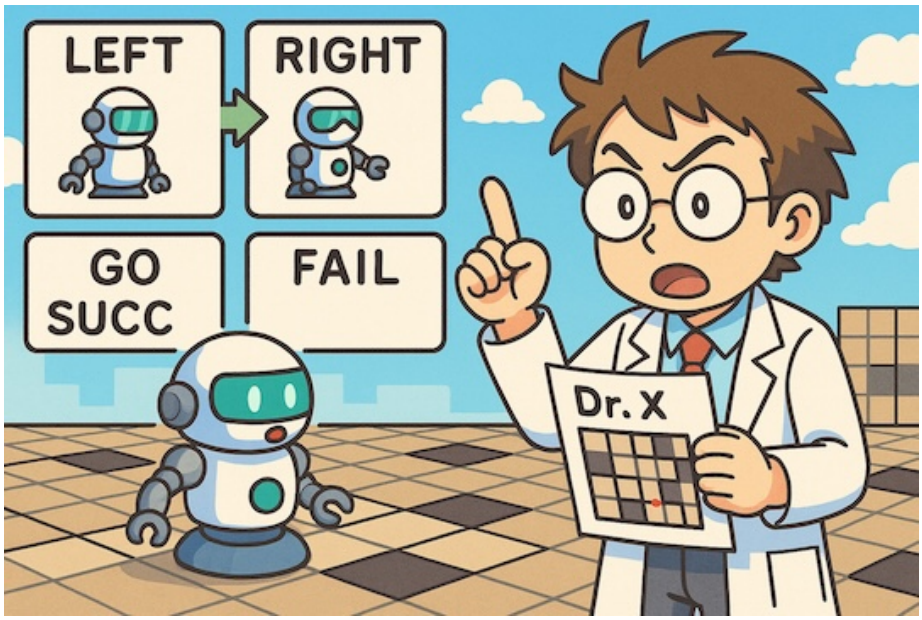
- 对于 40% 的数据，$n \leq 100$。
- 对于 100% 的数据，$10 \leq n \leq 10,000$，且输入中的所有成语均来自真实的成语词典。成语词典中没有重复的成语。

第五题 (p5.cpp, 100 分): 迷宫探险

Dr.X 把他的机器人超时空传送到了一个未知的空间，也不知道机器人的初始位置和方向。Dr.X 只知道这个未知的空间可以看作由方格组成的矩形方阵，其中的每个方格要么是障碍物，要么是空地。Dr. X 很想知道空间的结构，因此他打算让机器人探索这个空间，并绘制出迷宫的地图。

Dr. X 每次可以给他的机器人发送一条指令，并等待机器人反馈。指令有三种：

- LEFT，向左旋转 90 度。机器人会返回旋转后的方位 (E, W, S, N 分别表示东、西、南、北)。
- RIGHT，向右旋转 90 度。机器人会返回旋转后的方位 (E, W, S, N 分别表示东、西、南、北)。
- GO，尝试向前 (当前面前的方向) 移动到相邻的方格。如果前方的方格是障碍物，机器人会返回 FAIL。否则，机器人会返回 SUCC 并移动到到该方格。



你需要编程操纵 Dr. X 的机器人 (发送 LEFT/RIGHT/GO 指令)，读取机器人的反馈，并在探索完成后绘制出迷宫的地图。

输入输出格式

本题为交互题，你的程序通过标准输入输出 (键盘输入、屏幕输出) 与另一个程序交互：

```
while (true) {
    string feedback;

    command = ...;  // 确定下一条指令
```

```
    cout << command << endl;  // 输出一条指令
    cin >> feedback;  // 得到机器人的反馈

    if (all_explored) {  // 已经探明所有方格
        cout << "END" << endl;
        cout << n << " " << m << endl;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                cout << map[i][j];
            }
            cout << endl;
        }
        break;  // 退出循环
    } else {
        ...  // 继续探索
    }

}
```

即你的代码每次输出一行一条指令 (LEFT/RIGHT/GO)，读取机器人的反馈并逐步探测迷宫，直到全部可达的方格都被探明。探索完成后，程序输出 `END` 表示结束，随后输出迷宫的尺寸和地图，空地用点“`.`”表示，障碍物用井号“`#`”表示。

初始时，Dr.X 并不知道机器人所处的位置。你必须正确探明所有可达的格子，从机器人初始位置不可达的格子，统一认为是障碍物。

样例数据

其中“输出”代表你程序的屏幕输出 (机器人的命令)。“输入”是程序的键盘输入 (机器人的反馈)。在下面的例子中，机器人尝试过四个方向移动都失败后，绘制出仅有一个空格的迷宫地图。

```
(输出 cout) LEFT
(输入 cin)  N
(输出 cout) GO
(输入 cin)  FAIL
(输出 cout) LEFT
(输入 cin)  W
(输出 cout) GO
(输入 cin)  FAIL
(输出 cout) LEFT
(输入 cin)  S
(输出 cout) GO
(输入 cin)  FAIL
(输出 cout) LEFT
(输入 cin)  E
(输出 cout) GO
(输入 cin)  FAIL
```

```
(输出 cout) END
(输出 cout) 3 3
(输出 cout) ###
(输出 cout) #.#
(输出 cout) ###
```

数据规模

- 对于 50% 的数据，迷宫中可达的格子不超过 10 行、10 列。
- 对于 100% 的数据，迷宫中可达的格子不超过 100 行、100 列。你的程序调用 GO 的次数不得超过 50,000 次；你的程序输出的迷宫不得超过 102 行、102 列；未探测到的区域无需输出。

提示

你提交的代码存在交互式的循环：

```
cout << command << endl;  // 输出一条指令
cin >> feedback;  // 得到机器人的反馈
```

如果手工输入反馈，测试代码就会变得很困难。如果希望测试你的代码，我们建议将输入 (`cin >> feedback`) 替换为模拟实现的机器人反馈函数：

```
cout << command << endl;
feedback = get_feedback(command);
```

其中 `get_feedback` 函数在程序指定得到迷宫中模拟执行 `command` 命令并返回结果。**注意提交前务必将 `get_feedback` 替换回 `cin >> feedback`。**

第六题 (p6.cpp, 100 分): 程序套娃

套娃，原名“马特廖什卡娃娃”(Матрешка)，起源于 19 世纪末的俄罗斯，由若干逐渐变小、可以套进彼此的木制空心娃娃组成，象征家庭和生命的延续。套娃因其独特的艺术风格和寓意，逐渐成为俄罗斯民间艺术的代表和著名的旅游纪念品。

Dr. X 想到，程序能不能套娃呢？Dr.X 希望你写一个程序，输入整数 n 和 k，你的程序会输出一个字符串，**Dr. X 会把它保存为 p_1.cpp**，然后 Dr. X 会开始玩你的套娃程序：

- 首先，编译运行 p_1.cpp，得到它的屏幕输出，保存为 p_2.cpp。
- 在此基础上，编译运行 p_2.cpp，得到它的屏幕输出，保存为 p_3.cpp。
- 重复上述过程，编译运行 p_i.cpp，得到它的屏幕输出，保存为 p_{i+1}.cpp，直到得到 p_n.cpp 为止。
- Dr. X 期望 **p_n.cpp 不是一个 C++ 程序，而恰好是一个整数 k** (k 之后有无换行均可)。
- 在任何阶段，程序 p_1.cpp, p_2.cpp, ..., p_{n-1}.cpp 都只允许向标准输出 (屏幕输出) 打印，不允许使用操作系统功能，例如创建临时文件等。且这些程序都不得超过 100KB。

输入格式

输入一行两个整数 n 和 k。

输出格式

输出一个可编译的 C++ 程序，它在经历 n 次编译运行的流程后能输出一个整数 k。评测机会采用和你的环境完全相同的编译器 (Dev-Cpp 中的 GCC 4.9.2) 和编译选项编译 p_1.cpp, p_2.cpp, ..., p_{n-1}.cpp。

样例输入 1

```
1 99
```

样例输出 1

```
99
```

- 99 会被保存为 p_1.cpp。

样例输入 2

```
3 4096
```

样例输出 2

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

int main() {
    cout << "#include <iostream>" << endl;
    cout << "using namespace std;" << endl;
    cout << "int main() { cout << 4096 << endl; return 0; }" << endl;
    return 0;
}
```

- Dr. X 会把你程序的输出 (样例输出) 保存为 p_1.cpp。
- 编译运行 p_1.cpp，得到一个 `cout << 4096 << endl;` 的程序，保存为 p_2.cpp。
- 编译运行 p_2.cpp，得到 4096，保存为 p_3.cpp，符合 $n = 3, k = 4096$ 的要求。

数据规模

- 对于 40% 的数据，$n \leq 4$。
- 对于 100% 的数据，$2 \leq n \leq 8$, $k \leq 1,000,000,000$。注意**提交程序的大小限制为 100KB**。

提示

以下是几种输出一个双引号的方法：

- `cout << "\""`
- `cout << "\x22"`
- `putchar(34)`

类似地，你可以输出用 `"\\"` 或 `"\x5c"` 输出一个反斜杠。

小知识：任何一个“足够强大”(图灵完备) 的编程语言，都一定存在一种特殊程序，叫做“Quine”，它能在没有外部输入的情况下，分毫不差地输出自己的完整源代码——这被称为“不动点定理”。(本知识与解决本题无关。)