A 要结课啦!

时间限制：1000ms 内存限制：65536kb

题目描述

不知不觉，大家的C语言学习就要接近尾声了!

在这门课中，我们学习了以下的内容：

- C1，E1：基础运算
- C2，E2：基本数据处理
- C3，E3：程序结构
- C4，E4：函数
- C5，E5：数组
- C6，E6：指针
- C7，E7：结构与联合
- C8，E8：综合应用

现在，给你每个知识点所对应的C/E序号，请你输出该序号所对应的知识点。

输入

一个正整数，范围为[1,8]

输出

一行，为所对应的知识点。

输入样例

```
3
```

输出样例

```
程序结构
```

HINT

- 如果在本地中文乱码，别管，直接交!
- 这应该是我这学期第一次出水题QAQ

> 折磨大家一学期的C语言课，终于要结课了。
>
> 不知道大家在这一学期的学习中，都收获了什么。
>
> 可能你意外的发现，自己对编程十分感兴趣，那么欢迎你以后继续去深入的研究；
>
> 可能你最终意识到，编程是你自己似乎怎么都迈不过去的一个坎，也希望你不要气馁。
>
> 作为航类的学生，或许很多人都有这么一个疑问：
>
> “我又不是弄计算机的，我学这个有什么意义？”

> 但其实，在当今社会中，编程已经和数学，物理一样，成为了一个工科学生在学习各种学科时会用到的工具，会在你之后的研究与应用中起到重要作用。
>
> 当然现在，我们不用去思考太多，就在这门课的末尾，给自己一个美好的结局吧。
>
> 那些写过的所有AC，WA，TLE……的代码，
>
> 那些为C语言熬的每一个夜，
>
> 我们，
>
> 就此别过~

Author: Arthas

B 简单字符2.0

时间限制：1000ms 内存限制：65536kb

题目描述

`大眼睛猫猫`定义，在以下情况时，单词的大写用法是正确的：

- 全部字母都是大写，比如 "BUAA" ；
- 全部字母都是小写，比如 "accoding"；
- 首字母大写，其他字母小写， 比如 "Ceremony"

`大眼睛猫猫`给你一个字符串，如果大写用法正确，打印`true`；否则，打印`false`.

输入样例1

```
BUAA
```

输出样例1

```
true
```

输入样例2

```
LoveWillTearUsApart
```

输出样例

```
false
```

数据范围

字符串长度小于100，由小写和大写英文字母组成。

*Author:*从来不听后朋的*dwy*

C 冰雹猜想2022

时间限制：1000ms 内存限制：65536kb

题目描述

1976年的一天，《华盛顿邮报》于头版头条报道了这样一个故事：

70年代中期，美国各所名牌大学校园内，人们都像发疯一般，夜以继日，废寝忘食地玩弄一种数学游戏。这个游戏十分简单：任意写出一个自然数N，并且按照以下的规律进行变换：

如果是个奇数，则下一步变成3N+1。如果是个偶数，则下一步变成N/2。经过若干次循环后，人们发现，无论N是怎样一个数字，最终都无法逃脱回到谷底1。准确地说，是无法逃出落入底部的4-2-1循环，永远也逃不出这样的宿命。

"就像宇宙中的黑洞可以将任何物质，以及运行速度最快的光牢牢吸住，这个数学黑洞牢牢吸住了所有（非零）自然数。"

这就是著名的"冰雹猜想"。这个猜想至今无人证明，也无人推翻。

冰雹猜想的最大魅力在于其随机性与不可知性，其上下浮沉异常剧烈，比如自然数27，其变换过程（称为雹程）的顶峰值为9232，达到了原有数字的三百多倍。

本题借助C语言来模拟出一个正整数的雹程，并求出其变换过程中的最大值。

输入

输入共一行：

一个正整数N（保证数据范围在int类型内，且N不为1）。

输出

输出共两行：

第一行**倒序输出**正整数N的整个变换过程，即首元素为1，末元素为N的数列，各元素以空格相间隔。

（保证变换过程数据范围在long long范围内）

第二行输出变换过程中的最大值。

输入样例

```
17
```

输出样例

```
1 2 4 8 16 5 10 20 40 13 26 52 17
52
```

样例解释

逆序输出17的全部变换过程，其变换过程中最大值为52。
##

写在最后：

冰雹猜想蕴含了这样一种无奈的哲学：任何事物无论刚开始存在多么大的误差，最后都会自行修复。

这令我想起了王小波《万寿寺》中的结尾的一段话：

“长安城里的一切已经结束。一切都在无可挽回地走向庸俗。”

Author：LNB

D 301 Moved Permanently 2022

时间限制：300ms 内存限制：65536kb

题目描述

请根据输入的整数op和两个数a，b计算a+b

并将所有的输出数据重定向输出至文件301.txt

输入

输入为标准输入。多组数据输入，共 2n+1行；

第一行，一个整数 n ，表示数据组数；

接下来 $2n$ 行，每两行为一组数据。

前一行为一个整数op，表示接下来一行的数据类型，其中 1 表示 `int` ，2 表示 `long long`，3 表示 `double` 。后一行为两个数a，b，保证在上一行的数据类型可表示范围内。

为了避免出现不必要的精度错误，请不要使用其他的快速读入方式，**务必用 `scanf` 函数读入数据。**

输出

所有输出数据请重定向输出至文件 `301.txt`

对于每组数据输出一行，为a+b的结果，$double$ 类型保留 5 位小数。

输入样例

```
2
1
46 48
3
123.123 234.234
```

输出样例

```
94
357.35700
```

数据范围

$0 < n \leq 500$

保证输入的实数小数位数不超过 6 位。

整型相加结果保证在指定的数据类型的可表示范围内

测评结果解释

本题采用 SPJ ，测评结果可能与其他题目不同，以下为可能遇到的非常见测评结果。

OFNR（Output File Not Ready）：输出文件错误，请仔细阅读输出要求。

IFNR（Input File Not Ready）：测评程序错误，请联系助教。

OE（SPJ Error）：测评程序错误，请联系助教。

E 简单的种田

时间限制：1000ms 内存限制：65536kb

题目介绍

由于基金雪崩，JJJ以后可能要去种田了。

田地是一个巨大的矩形，由于JJJ掌握了核心科技，所以他每次**能且只能**种一个正方形，而每种一个正方形时JJJ所花的体力值是正方形的**周长**。JJJ很懒，而且他还要节约体力去出题，所以他想花最少的体力值种完这块田地，请问最小的体力值是多少。

特别地，JJJ种过的田不能够再种。

输入格式

两个正整数x, y， 表示田地的长和宽。

输出格式

一个正整数，表示最小的体力值。

样例输入

```
2 3
```

样例输出

```
16
```

数据范围

$1 \leq x, y \leq 10^{16}$.

答案保证在$long\ long$范围内。

HINT

递归

如果TLE了，可以想想有没有能够优化的地方。

Author: ///

F 蒙达鲁克硫斯伯古比奇巴勒城的名册2.0

时间限制：1000ms 内存限制：204800kb

题目背景

`蒙达鲁克硫斯伯古比奇巴勒城`住着很多居民，他们的名字都很长，比如王子`达拉崩吧斑得贝迪卜多比鲁翁`、公主`米娅莫拉苏娜丹妮谢莉红`和皇孙`王浩然`。

现给出每个人的年龄和名字，请根据输入的顺序进行 m 次操作：每次操作将区间 $[l,r]$ 中的人的顺序翻转，最后对于 m 次操作之后，输出 k 次查询的结果

输入格式

第一行三个整数 n,m,k ，分别表示总共有多少人、操作的次数和查询的次数；

接下来 n 行每行一个整数 a 表示年龄和一个字符串s (仅由英文小写字母组成) 表示名字，用一个空格分开。

接下来 m 行，每行两个正整数 l,r ，表示将区间 $[l,r]$ 中的人的顺序翻转（从1开始数）；

接下来 k 行每行一个整数,表示需要查询上述操作完成后第几个人的信息（从1开始数）。

输出格式

对于每次查询，输出一行，包含一个整数和一个字符串，用空格分开，表示操作后该位置的人的年龄和名字。

输入样例

```
3 1 2
2 aaa
1 bbb
2 ccc
1 2
1
3
```

输出样例

```
1 bbb
2 ccc
```

样例解释

输入顺序为 `2 aaa`,`1 bbb`,`2 ccc`，经过一次翻转操作，将区间 [1,2] 内的顺序翻转之后的顺序为 `1 bbb`,`2 aaa`,`2 ccc`，故对于查询1和3，输出如上

数据范围

$1 \leq n \leq 200005$

$1 \leq m \leq 100$

$1 \leq k \leq 10$

$0 \leq a \leq 10000$

$1 \leq len(s) \leq 1000$

HINT

如果对结构体直接交换不行的话，试试构建一个相应的结构体指针数组，然后对指针交换，而不直接对结构体交换（想想为什么这样可行）。

AUTHOR:dch，cbd

G 灭火

时间限制：1000ms 内存限制：65536kb

题目介绍

消防站里警铃大作！接到火警：一排住宅楼中有几栋楼房发生了火灾！消防员们迅速出动，同时需要你帮助他们判断火势蔓延的速度，请抓紧时间吧！

已知有 N 座楼房并排位于街道一侧，其中有 M 座发生了火灾，**第 1 分钟**仅在原所在的楼宇开始燃烧，接下来每一分钟，一座楼房的火灾就会蔓延到左右两座楼房；假设每座楼房的火势不会自动熄灭，请问多久火灾会蔓延到所有楼房。

输入格式

输入共两行；

第一行两个用空格分隔的正整数 N, M ，表示一共有 N 座并排的楼房，其中有 M 座发生了火灾；

第二行 M 个两两不同的、用空格分隔的正整数 k_i ，$1 \leq k_i \leq N, i = 1, 2, \cdots, M$ ，表示初始时刻第 k_i 座楼房发生了火灾；

输出格式

输出一行一个正整数 T ，表示在第 T 分钟，火势首次蔓延到所有楼房；

输入样例1

```
5 1
3
```

输出样例1

```
3
```

输入样例2

```
3 3
2 3 1
```

输出样例2

```
1
```

样例解释

样例一中，第 1 分钟着火的楼房有 3；第 2 分钟着火的楼房有 $2, 3, 4$；在第 3 分钟，着火的楼房有 $1, 2, 3, 4, 5$ ，首次所有楼房都着火了，故输出 3；

样例二中，在第 1 分钟所有楼房就已经着火，故输出 1；

数据范围

对于 40%的数据，$1 \leq N \leq 10^3$；

对于 60% 的数据，$1 \leq N \leq 10^5$；

对于 100% 的数据，$1 \leq N \leq 10^6$ 。

AUTHOR：*Bodan Chen*

H 找内鬼2.0

时间限制：1000ms 内存限制：65536kb

题目背景

内鬼又回来了！在5.20号这天 TA（也许）带上了 TA 的npy。

简单描述

在输入的 n 个数中**至多**有2个数出现了奇数次，其他数字都出现了偶数次，请输出出现了奇数次的数；如果所有数字都出现了偶数次，输出`False Alarm.`

输入

共2行。

第一个数为输入数据的总数 n。

接下来一行共有 n 个整数，每一个整数以空格隔开，代表课程内的编号或点名册上的编号。由于课程内有老师同学和教职工等，因此编号的位数并不统一。

输入时并不按照“课程内编号——点名册" 的顺序输入（也就是说是乱序输入哒！）。

输出

输出一行整数，表示内鬼伪装成的编号，若有两个内鬼，则按**从小到大**的方式输出；若实际上没有内鬼的话， 输出`False Alarm.`。

保证内鬼最多只会有两个。

输入样例1

```
8
2345 34567 34567 456789 456789 2345 456789 2345
```

输出样例1

```
2345 456789
```

输入样例2

```
4
1234 2345 1234 2345
```

输出样例2

```
False Alarm.
```

数据范围

保证输入数据总数n满足$1 \leq n \leq 10^6$，对每一个编号a_i，有$1 < a_i < 10^9$。

HINT

如果你忘了怎么通过位运算找一个内鬼，可以点击这里。

存在两个内鬼的时候，他们的值一定不同，因此二进制表示也一定不同。我们知道，对两个不同的数进行按位异或运算时，不同的位上运算结果为1。此时，假设异或结果是：

$$0\,0\,0\,0\,0\cdots0\,1\,0\cdots0\,0$$

$$\uparrow$$

在箭头所指处，两个内鬼表示的值在这一位上的值一定不相同。也就是说，若我们假设内鬼 **a** 在这一位为1，内鬼 **b** 这一位则肯定为0。带着这个新的信息，我们重新对原数据进行一些处理，便可以得到答案。

记得对答案排序噢~

Author: czy

I 数据类型

时间限制：1000ms 内存限制：65536kb

题目描述

请判断输入对应的数据类型，可能为 `int` 、 `long long` 、 `string（字符串）` 、 `double` 。

如果输入的内容为小数，则应输出 `double` ； 如果输入的内容为整数且在 `int` 范围内，输出 `int` ； 如果输入内容为整数，不在 `int` 范围内但在 `long long` 范围内，输出 `long long` ； 如果输入的内容为整数，但不在 `long long` 范围内，则输出 `string` ； 如果输入内容为字符串，则输出 `string` 。

输入

多行输入。

每行输入不超过30字符，且仅包括小写英文、数字和小数点。

输出

每行输入对应一行输出，输出对应的数据类型即可。

输入样例

```
123456
888888888888888888
123.456
asdfghjkl
```

输出样例

```
int
long long
double
string
```

Hint

AUTHOR：爱吃猪脚的猪脚

J rlx不教你写代码（二）

时间限制：1000ms 内存限制：65536kb

题目描述

代码风格（英语：Programming style）即程序开发人员所编写源代码的书写风格。良好代码风格的特点是使代码易读。（摘自维基百科）

遵守良好的代码风格也许不能帮助你AC题目，但是在工程领域中，拥有良好的代码风格甚至遵守一定的代码规范是开发工程师的必备素养。一份拥有良好代码风格的代码可以帮助开发者及其后人更好地维护代码。形象地说，代码的风格就如同一个人的字写得是否好看。

在C语言中常见的良好书写风格有：正确缩进、恰当添加空格、起有意义的变量名和添加必要的注释等。

现在给予你一份语法完全正确但格式不规范的C语言代码，请你将其格式化为规范书写的代码。给予你的代码中可能包含的语法元素以及具体的代码格式化要求如下：

- 关键字，包括：

  ```
  int, long, short, float, double, char, unsigned, signed, const, void,
  if, else, do, while, for,
  continue, break, sizeof, return
  ```

  其中`continue`，`break`，`sizeof`的两侧没有空格；`return`后如果直接是分号，则两侧没有空格，否则只有右侧有一个空格；其余关键字只有右侧有一个空格，除非该关键字在小括号内用作强制类型转换，此时关键字两侧没有空格。
- 运算符，包括：（所有可能出现的运算符见题目最后的**附录**）
  - 二元运算符：两侧有且仅有一个空格，下方给出所有可能出现的二元运算符；

    ```
    * / % + - << >> < <= > >= == != & ^ | && || = += -= *= /= %= <<= >>= &=
    ^= |=
    ```
  - 三元运算符：`?`和`:`的两侧有且仅有一个空格；
  - 逗号：`,`只有右侧有一个空格；
  - 其余的都视为一元运算符，且两侧没有空格。
- 合法命名的标识符，两侧没有空格。
- 常量，包括字符常量、字符串常量、整型常量（无前后缀）和浮点型常量（无尾缀、小数点前后都至少有一位数字）。常量的两侧没有空格。
- 预处理命令，只可能是`#include`，只有`#include`的右侧有一个空格，`#include`后的内容两侧没有空格。
- 用于确定运算优先级的小括号`()`，`(`和`)`的两侧没有空格。
- 分号`;`，当分号在`for`关键字后的括号中时，分号仅有右侧有一个空格；其余情况下分号仅有右侧有一个`\n`。
- 大括号`{}`，`{`和`}`的两侧有且仅有一个`\n`，只有当`}`的后一个非空白字符为`,`或`;`时，`}`的后面没有`\n`。并且同一**大括号对**内的代码为一个代码块，此代码块比大括号所在的代码块级别高**1**。代码块中的每一行代码前应该有**级别乘以4**个空格。起始代码块的级别为**0**。保证所有的代码块都有**大括号对**包围。

- 没有任何形式的注释。

输入

多行字符串，一份语法完全正确但格式不规范的C语言代码，所包含的内容如题目规定。代码大小不超过20KB，每行字符串的长度不超过1000。保证输入代码中的空白符只包括`\n`和空格。

输出

多行字符串，代表按要求格式化后的代码。

输入样例1

```
#include <stdio.h>
int months[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
int isleap(int y) {
 if ((y % 400 == 0) || (y % 4 == 0 && y % 100 != 0))
 {
 return 1;
 }
 return 0; }
int main()
{
 int n,y,d;
 scanf("%d",&n);
 for(int i=0;i<n;i++)
 {
 scanf("%d%d",&y,&d);

 if(isleap(y)){months[2]=29;}
 else{ months[2]=28;
}

 int mon=1;
 while(d>months[mon])
 {
 d-=months[mon];
 mon++;
 }
 printf("%d %d\n",mon,d);
 } }
```

输出样例1

```
#include <stdio.h>
int months[13] =
{
    0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
};
int isleap(int y)
{
    if ((y % 400 == 0) || (y % 4 == 0 && y % 100 != 0))
    {
        return 1;
    }
```

```
    return 0;
}
int main()
{
    int n, y, d;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d%d", &y, &d);
        if (isleap(y))
        {
            months[2] = 29;
        }
        else
        {
            months[2] = 28;
        }
        int mon = 1;
        while (d > months[mon])
        {
            d -= months[mon];
            mon++;
        }
        printf("%d %d\n", mon, d);
    }
}
```

输入样例2

```
#include<stdio.h>
#include<string.h>
#include<math.h>

void _do_some_thing(int param1,double param2, char*param3[]) {
long long just_a_variable=1;
just_a_variable<<=12;just_a_variable>>=11;just_a_variable=1<<2;
int array_1[][10]={{1,2},{3,4},{5,6,7}};
double *ptr;
double val1=1.0;val1+=1.0,ptr=&val1;
int a_long_expression=(-1.0/2)*(2+((int)(-val1-(long long)-val1-*ptr)*
(&array_1[1][-2*2+10]-&array_1[1][1])&array_1[2][2]));
a_long_expression = a_long_expression>just_a_variable?(param1-=1):(param2*=2);
just_a_variable^=just_a_variable;

if(param1&1&&(float)param2<0) {
do{printf("\"Hello\":%s",param3[just_a_variable]),++just_a_variable;}while(param
3[just_a_variable]!=NULL);
} else if(param1&1){
param1|=2*-param1;
param1*=2;
if ((param1<<1)|1) {
param1/=2;

param1-=1;
if (param1>>1) {
    param1>>=4;
} else {      param1<<=5;}
```

```
int x0,*x1,**x2,***x3,****x4,*****x5;
x1=&x0;x2=&x1;x3=&x2;x4=&x3;x5=&x4;
x1++,++x2,++x3,x4--,--x5;
x0=-x0+(-1+1&*x1***x2)/(((***x3%****x4)-'\'')>>*****x5<<(*x1>=**x2 && -
x0<-1234));
while (param1>x0||(param2<=*x1&&fabs(param2-***x3)<0.000001)) {
    param1 %= -2;param1 &= 2;
    param1 >>= *x1;param1<<=**x2;
}
} }
return;
}

#include  <stdlib.h>
int cmp(const void*a, const void*b) {
long long int na = *(long long*)a,nb=*(long long*)b;
if(na>nb){return 1;}


else if(na <nb){return -1;}
else{return 0;}
}


long long int n,a[10007];
char *languages[]={"C","C++","Python","Java","Go"};
int main(int argc, const char *  argv[])
{

scanf("%lld",&n);
for (int i = 0; i < n; i+=2) {
scanf("%lld %lld",&a[i],&a[1+i]);
}
qsort(a,  n,     sizeof(long long int),  cmp);
for (int i = 0; i < n; ++i) {
printf("%c",'A'+(unsigned)a[i]-'a');
}
_do_some_thing(-1, 3.14*114.514, languages);
return 0;
}
```

输出样例2

```
#include <stdio.h>
#include <string.h>
#include <math.h>
void _do_some_thing(int param1, double param2, char *param3[])
{
    long long just_a_variable = 1;
    just_a_variable <<= 12;
    just_a_variable >>= 11;
    just_a_variable = 1 << 2;
    int array_1[][10] =
    {
        {
            1, 2
        },
```

```
        {
            3, 4
        },
        {
            5, 6, 7
        }
    };
    double *ptr;
    double val1 = 1.0;
    val1 += 1.0, ptr = &val1;
    int a_long_expression = (-1.0 / 2) * (2 + ((int)(-val1 - (long long)-val1 -
*ptr) * (&array_1[1][-2 * 2 + 10] - &array_1[1][1]) & array_1[2][2]));
    a_long_expression = a_long_expression > just_a_variable ? (param1 -= 1) :
(param2 *= 2);
    just_a_variable ^= just_a_variable;
    if (param1 & 1 && (float)param2 < 0)
    {
        do
        {
            printf("\"Hello\":%s", param3[just_a_variable]), ++just_a_variable;
        }
        while (param3[just_a_variable] != NULL);
    }
    else if (param1 & 1)
    {
        param1 |= 2 * -param1;
        param1 *= 2;
        if ((param1 << 1) | 1)
        {
            param1 /= 2;
            param1 -= 1;
            if (param1 >> 1)
            {
                param1 >>= 4;
            }
            else
            {
                param1 <<= 5;
            }
            int x0, *x1, **x2, ***x3, ****x4, *****x5;
            x1 = &x0;
            x2 = &x1;
            x3 = &x2;
            x4 = &x3;
            x5 = &x4;
            x1++, ++x2, ++x3, x4--, --x5;
            x0 = -x0 + (-1 + 1 & *x1 * **x2) / (((***x3 % ****x4) - '\'') >>
*****x5 << (*x1 >= **x2 && -x0 < -1234));
            while (param1 > x0 || (param2 <= *x1 && fabs(param2 - ***x3) <
0.000001))
            {
                param1 %= -2;
                param1 &= 2;
                param1 >>= *x1;
                param1 <<= **x2;
            }
        }
    }
```

```
    return;
}
#include <stdlib.h>
int cmp(const void *a, const void *b)
{
    long long int na = *(long long *)a, nb = *(long long *)b;
    if (na > nb)
    {
        return 1;
    }
    else if (na < nb)
    {
        return -1;
    }
    else
    {
        return 0;
    }
}
long long int n, a[10007];
char *languages[] =
{
    "C", "C++", "Python", "Java", "Go"
};
int main(int argc, const char *argv[])
{
    scanf("%lld", &n);
    for (int i = 0; i < n; i += 2)
    {
        scanf("%lld %lld", &a[i], &a[1 + i]);
    }
    qsort(a, n, sizeof(long long int), cmp);
    for (int i = 0; i < n; ++i)
    {
        printf("%c", 'A' + (unsigned)a[i] - 'a');
    }
    _do_some_thing(-1, 3.14 * 114.514, languages);
    return 0;
}
```

HINT

关键词：词法分析，Allman风格

有的二元运算符在某些情况下会变成一元运算符。

测试数据跟样例差不多捏。本题以体验为主，有兴趣的同学可以将功能补全后，形成自己风格的代码格式化工具。

Author: Lucien Li

附录

可能出现的所有运算符：

```
++ --       后缀自增与自减
()      函数调用
```

```
[]      数组下标
++ --       前缀自增与自减
+ -    一元加与减
! ~    逻辑非与逐位非
(type)      转型
*       间接（解引用）
&       取址
* / %      乘法、除法及余数
+ -        加法及减法
<< >>      逐位左移及右移
< <=       分别为 < 与 ≤ 的关系运算符
> >=       分别为 > 与 ≥ 的关系运算符
== !=      分别为 = 与 ≠ 关系
&       逐位与
^       逐位异或（排除或）
|       逐位或（包含或）
&&          逻辑与
||      逻辑或
?:      三元条件
=       简单赋值
+= -=      以和及差赋值
*= /= %=   以积、商及余数赋值
<<= >>=    以逐位左移及右移赋值
&= ^= |=   以逐位与、异或及或赋值
,       逗号
```

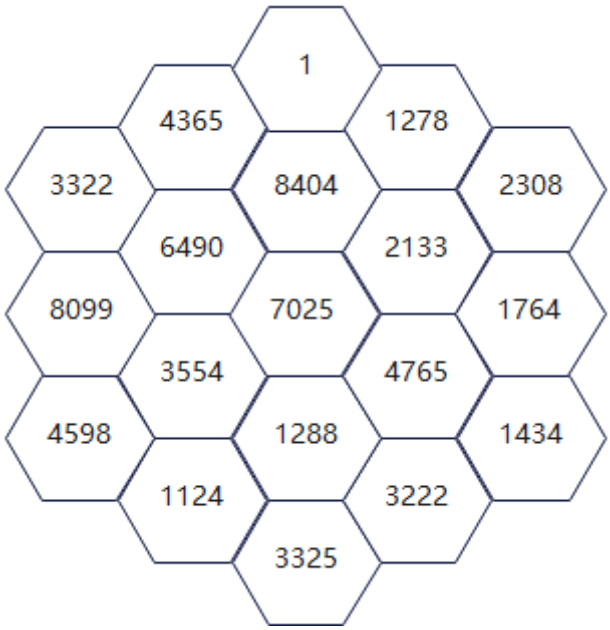
K 六宫数局

时间限制：1000ms 内存限制：65536kb

本题为期中考试题，限时开放

题目描述

我们之前遇到过一个名叫方阵数局的题目。这次，我们将数字阵列的形状改变为蜂巢型，如图所示：

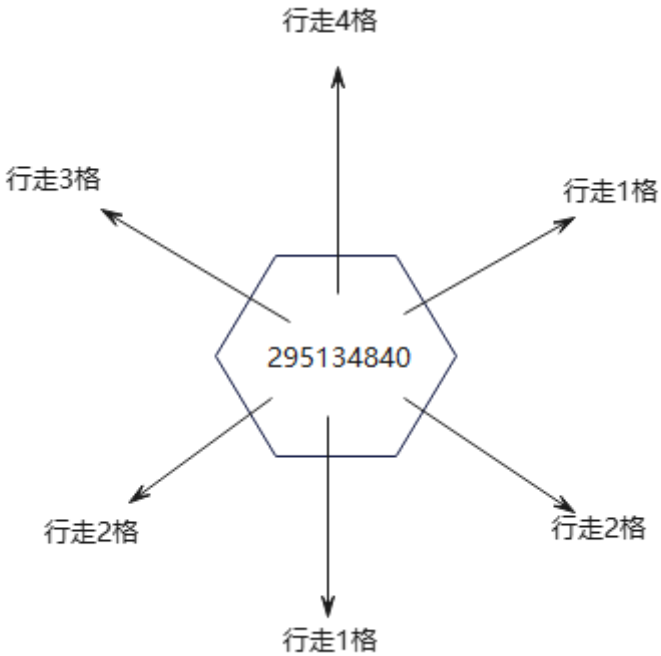


你的目标是，从最下方的六边形开始，按照一定规则，走到最上方的六边形，规则如下：

1. 将六边形中的数分解质因数，如

$$295134840 = 2^3 \times 3^4 \times 5 \times 7^2 \times 11 \times 13^2$$

2. 将自然数中最小的六个质数$2, 3, 5, 7, 11, 13$分别代表左上，正上，右上，右下，正下，左下。
3. 六边形中的数分解出这六个数的次数代表从该位置开始能在这六个方向行进的格数（**注：例如分解出的次数为3，则只能走3格，而不能只走一格或两格**），如图所示：



4. 每个点只能停留一次（被横跨的点不算停留）

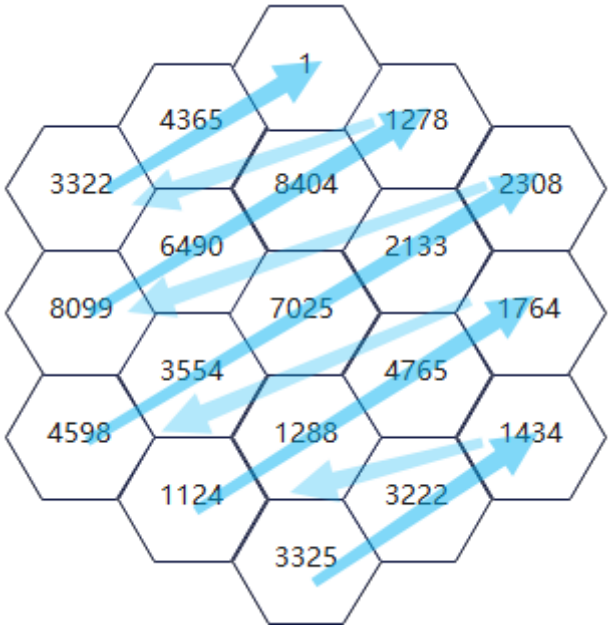
请问，你能否找到一条路径，破解六宫数局。

输入

第一行输入一个正整数n，为六边形外侧边的长度（示意图中为3）。

接下来$2n-1$行：

分别输入$n, n+1, \ldots, 2n-1, \ldots n+1, n$个数，铺满六边形，铺设顺序如下图：



输出

输出若干行，格式为`方向 步长`，表示行走的路径：

方向表示方法为：

方向	表示
左上	LU
正上	U
右上	RU
右下	RD
正下	D
左下	LD

输入样例

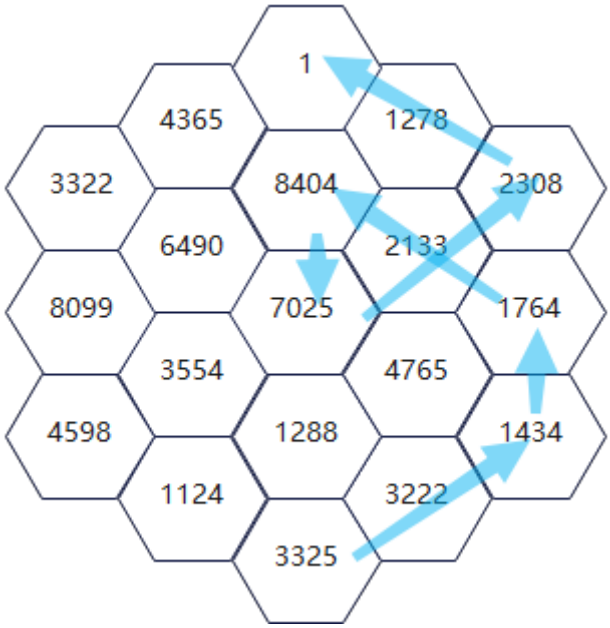
```
3
3325 3222 1434
1124 1288 4765 1764
4598 3554 7025 2133 2308
8099 6490 8404 1278
3322 4365 1
```

输出样例

```
RU 2
U 1
LU 2
D 1
RU 2
LU 2
```

样例解释

路径如图所示



数据范围

$1 < n <= 7$;

阵列中的数为`int`范围内的正整数;

保证至少有一种可行通路;

可行通路可能不止一条，输出一条即可;

Author: Arthas