2022 “信息与未来” 小学生夏令营编程活动

注意事项

- 请遵照机房老师的指示建立目录、保存文件和提交。
- 使用标准输入输出 (即键盘输入、屏幕输出，不需要打开输入/输出文件)。
- 用 C++ 语言答题。请提交你解答问题的源代码文件 (六个题目的源文件名分别为 p1.cpp, p2.cpp, p3.cpp, p4.cpp, p5.cpp, p6.cpp。请严格按此要求命名)。大小超过 100 KiB 的代码不予评测。
- 采用全自动评测。测试点运行时使用超过 256 MB 内存或 1s 时间的程序将被立即终止 (该测试点不得分)。评测机为主流桌面 Intel 处理器。

第一题 (p1.cpp, 15 分): 幸运数字

如果一个十进制数字从左到右读时，所有数位都是从小到大严格递增的，我们就称它是幸运数字。例如：

- 9、27、1234 都是幸运数字；
- 11、80、243、1503 不是幸运数字。

对于给定的 a 和 b，请你求出 $a, a+1, a+2, \ldots, b$ 中幸运数字的数量。

输入格式

输入一行空格分隔的两个整数 a 和 b。

输出格式

输出一行一个整数，代表 $a, a+1, a+2, \ldots, b$ 中幸运数字的数量。

样例输入 1

```
1 100
```

样例输出 1

```
45
```

样例输入 2

```
4096 65536
```

样例输出 2

```
141
```

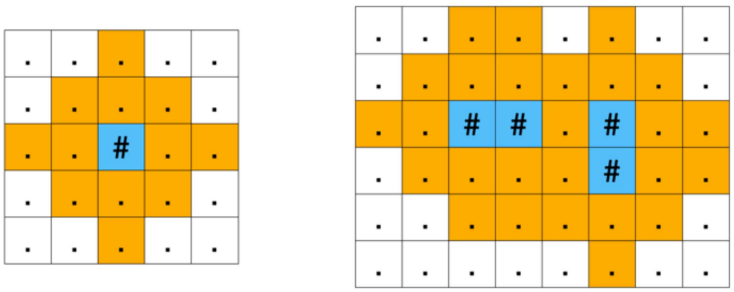
数据规模

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

第二题 (p2.cpp, 15 分): 沙滩面积

我们获得了一幅陆地和水域情况的卫星照片，照片可以看成是一个 n 行 m 列的矩形，矩形中的每个格子要么是陆地（用半角点号 . 表示），要么是水域（用井号 # 表示）。

虽然卫星照片可以清楚地辨别出陆地和水域，但陆地的具体类型却并不明确。现在我们了解到，对于一块是水域的方格，它向上、下、左、右四个方向 k 步之内可达的陆地格子，均会形成沙滩。例如，下图展示了 $k = 2$ 的情况，蓝色的格子代表水域，标为黄色的陆地格子是沙滩：



你的任务是根据卫星照片计算出属于“沙滩”格子的数量。注意：卫星照片只拍摄了包含水域的部分，水域附近的沙滩可能出现在卫星照片边界之外。你可以假设卫星照片之外不存在任何水域。

输入格式

输入的第一行是空格分隔的三个整数 n, m 和 k，代表拍摄的卫星照片共有 n 行 m 列，以及形成沙滩的范围 k。

接下来 n 行，每行一个字符串。字符串的长度恰好是 m，代表卫星照片的一行，其中：

- 井号 # 表示一片水域；
- 半角点号 . 表示一片陆地。

输出格式

输出一行一个整数，代表沙滩格子的数量。

样例输入 1

```
2 4 2
##.#
...#
```

样例输出 1

```
26
```

样例输入 2

```
5 10 3
.........#
..########
.........#
#....###.#
..####...#
```

样例输出 2

```
103
```

数据规模

- 对于 40% 的数据，满足 $n = m = 1$；
- 对于 100% 的数据，满足 $1 \leq n, m \leq 100; 1 \leq k \leq 10$。

第三题 (p3.cpp, 15 分): 完美字符串

你可能见过下面这一句英文：

> “The quick brown fox jumps over the lazy dog.”

短短的一句话就包含了所有 26 个英文字母！因此这句话广泛地用于字体效果的展示。更短的还有：

> “The five boxing wizards jump quickly.”

所以你很好奇：还有没有更多这样包含所有 26 个英文字母的句子？于是你用爬虫在互联网上爬取了许多英文文本，并且提取出了其中的单词。你现在希望从一个很长的单词序列中找出一段连续出现的单词，它满足：

- 所有 26 个英文字母都至少出现一次；
- 长度尽可能短，即包含的字母总数尽可能少。

输入格式

输入的第一行包含一个整数 n，代表单词序列的长度，即单词的数量。

输入的第二行包含 n 个空格分隔的英文单词（**单词仅由小写字母构成**）。输入数据保证每个小写英文字母都至少出现一次。

输出格式

输出一行一个整数，是你找到的单词序列中的字母总数。

样例输入

```
13
there is a quick brown fox jumping over the lazy dog and cat
```

样例输出

```
37
```

最短满足条件的单词序列是“is a quick brown fox jumping over the lazy dog”，共有 37 个字母。

数据规模

- 对于 40% 的数据，满足 $n \leq 100$；
- 对于 100% 的数据，满足 $1 \leq n \leq 100,000$。每个单词的长度不超过 10 个字符，且单词全部由小写英文字母 a-z 构成。

第四题 (p4.cpp, 15 分): 样本分类

在人工智能中，“分类”是一项非常重要的任务。例如，公安系统通过人工智能分类潜在的电信诈骗，并进行及时的预警。现在我们考虑一种最简单的“线性二分类器”，它由 n 个整数参数（常数）$a_1, a_2, \ldots a_n$ 组成。在输入一组整数 $x_1, x_2, \ldots, x_n$ 时，如果

$$a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots + a_n \cdot x_n \leq 0$$

成立，则分类器输出 true，否则分类器输出 false。如果分类器的输入（称为“特征”）选择得当，即便是简单的线性分类器也能取得很好的分类效果。

现在，我们得到了两个用于同一分类任务的分类器：

- 分类器 A，当 $a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots + a_n \cdot x_n \leq 0$ 时输出 true，否则输出 false
- 分类器 B，当 $b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_n \cdot x_n \leq 0$ 时输出 true，否则输出 false

由于 A 和 B 是两个不同的分类器，它们可能会在某些输入的 x 上产生分歧，给出不同的分类结果——这样

的问题值得我们特别关注。因此，你的任务就是求出一组 $x_1, x_2, \ldots, x_n$，使得分类器 A 和 B 返回不同的结果。

为了帮助大家理解，我们假设 a_i 和 b_i 分别保存在数组 a 和 b 中（数组下标从 1 到 n）。你的任务就是求出数组 x 的元素 x[1], x[2], ..., x[n]，使下面的函数 diverge 返回 true：

```
int n; // 参数/输入的个数

bool classify(int param[], int x[]) {
  int y = 0;
  for (int i = 1; i <= n; i++) {
    y += param[i] * x[i];
  }
  if (y <= 0) {
    return true;
  } else {
    return false;
  }
}

bool diverge(int x[]) {
  return classify(a, x) != classify(b, x);
}
```

输入格式

输入的第一行包含一个整数 n，代表分类器参数的个数。

输入的第二行包含 n 个整数，代表分类器 A 的参数 a_i $(1 \leq i \leq n)$。

输入的第三行包含 n 个整数，代表分类器 B 的参数 b_i $(1 \leq i \leq n)$。

输入数据保证两个分类器的参数不是完全相同的。

输出格式

输出一行 n 个整数，分别是 $x_1, x_2, \ldots, x_n$。其中 x_i $(1 \leq i \leq n)$ 可以是正数、负数或零，但绝对值不能超过 10,000。如有多组满足要求的解，输出任意一个即可。

样例输入

```
2
1 1
-1 1
```

样例输出

```
100 0
```

数据规模

- 对于 100% 的数据，满足 $1 \le n \le 1,000$，$-100 \le a_i, b_i \le 100$ $(1 \le i \le n)$。

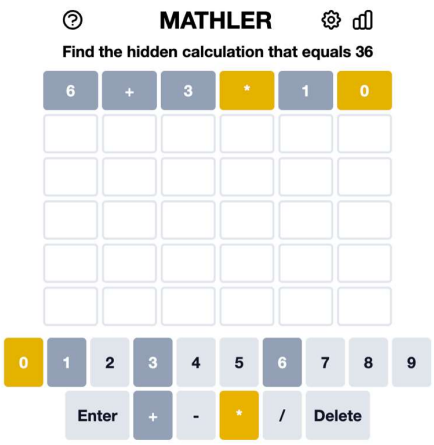
第五题 (p5.cpp, 20 分): Mathler

风靡全球的猜单词游戏有了数学版本！在这个游戏中，计算机会秘密产生一个隐藏的表达式，例如 $90/5 \times 2 = 36$。但计算机不会把具体的表达式告诉你，而是只把表达式的计算结果 (在前面的例子中是 36) 告诉你。接下来，你需要用尽可能少的次数把这个表达式猜出来！

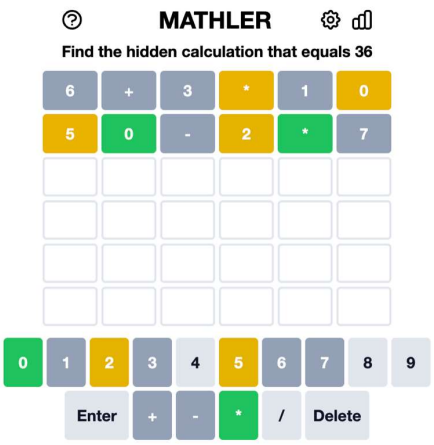
你可以和计算机多次互动完成猜测。在游戏的每一轮中，你都可以告诉计算机一个长度是 6 的字符串表达式，仅由数字 `0123456789` 和四则运算 `+-*/` 构成，**表达式中不含负数且数字不包含前导零**，并且**表达式的计算结果必须是游戏开始时给出的数值** (在我们的例子中是 36)。计算机收到你猜测的表达式后，会对你给出表达式中的每一个字符分别给出反馈：

- 绿色：这个字符出现在隐藏表达式中，且位置正确；
- 黄色：这个字符出现在隐藏表达式中，但位置不正确；
- 灰色：这个字符没有出现在隐藏表达式中。

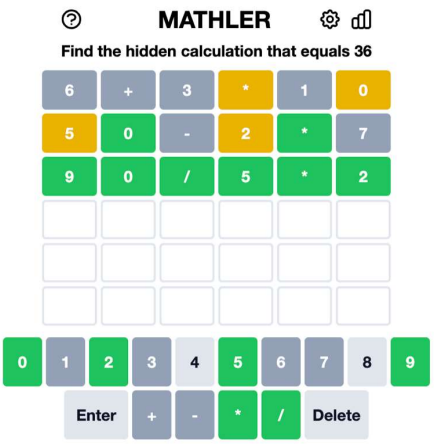
下面是一个三次猜中隐藏表达式的例子：



$6 + 3 \times 10 = 36$
0 和乘号出现在最终结果中
但不在正确位置 (黄色)



$50 - 2 \times 7 = 36$
0 和乘号出现在正确位置 (绿色)
2 和 5 不在正确位置 (黄色)



$90/5 \times 2 = 36$
所有数字均在正确位置 (绿色)
游戏结束

在你猜了很多次以后，觉得这个游戏已经有些无聊了。现在你希望编一个程序，让你的程序代替你在 12 次猜测之内得到隐藏的表达式。

输入输出格式

本题为交互题，评测时会模拟键盘输入并读取程序的屏幕输出。首先，评测程序会在键盘输入一行一个整数 n，代表隐藏表达式的计算结果，然后进入交互模式并等待你的程序输出一行 (对隐藏的表达式作出猜测)。评测程序读取到你屏幕上的输出后，会及时用键盘输入的方式给出反馈。以下 C++ 代码片段可以实现与评测程序的交互 (你需要补全其他必要的内容)：

```
cin >> n; // int n;
while (true) {
  cout << guess(n) << endl; // 作出一次猜测，注意需要输出一个换行
  cin >> res; // string res; 计算机对猜测的反馈
  if (res == "AAAAAA") {
    break; // 游戏结束
  }
}
```

在交互过程中，评测程序会及时的对你的输出给出反馈（作为键盘输入给程序）：

- 对于不正确的表达式（例如有多余的空格、表达式不合法或值错误等），反馈六个半角减号（“-”）；
- 对于值正确的表达式，反馈六个字符，其中：
  - “A”表示在表达式中出现且位置正确；
  - “B”表示在表达式中出现且位置不正确；
  - “X”表示在表达式中未出现。

注意隐藏表达式中同一个数字可能多次出现。如果你猜测的表达式中某个数字出现了多次，评测程序会优先给出“A”的结果，然后从左到右给出“B”的结果，多余的数字给出“X”的结果。在 12 次猜测以内得到“AAAAAA”且程序正常终止即被判定为正确。

本题中四则运算规则：先乘除、后加减，同优先级从左到右计算，允许计算过程中出现负数。除法是有理数的除法而非整除，即 `2/5*10` 的值为 4 且禁止除零。完整的交互过程应在 1 秒内完成；评测程序的运行时间很短，可以忽略不计。

样例输入/输出 1

```
36          (键盘输入)
6+3*10      (屏幕输出)
XXXBXB      (键盘输入)
6+3*11      (屏幕输出，表达式的值不等于36)
------      (键盘输入，半角减号，提示输出不合法)
50-2*7      (屏幕输出)
BAXBAX      (键盘输入)
90/5*2      (屏幕输出)
AAAAAA      (键盘输入，游戏结束，程序退出，共猜4次)
```

上面的交互过程对应了题图的交互过程，但中间有一次输入非法表达式的情况。

样例输入/输出 2

```
8           (键盘输入)
10-4+2      (屏幕输出)
BXXXXX      (键盘输入)
7*6-34      (屏幕输出)
XXXXBX      (键盘输入)
3+15/3      (屏幕输出)
ABBABX      (键盘输入)
```

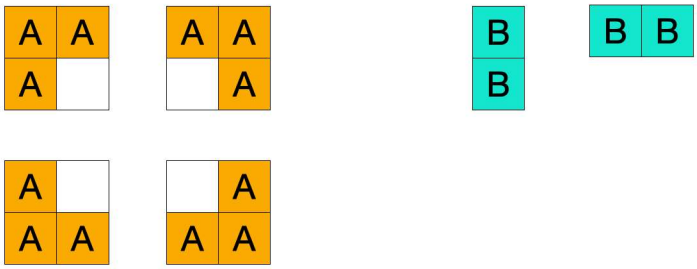
```
35/5+1      （屏幕输出）
AAAAAA      （键盘输入，游戏结束，程序退出，共猜4次）
```

数据规模

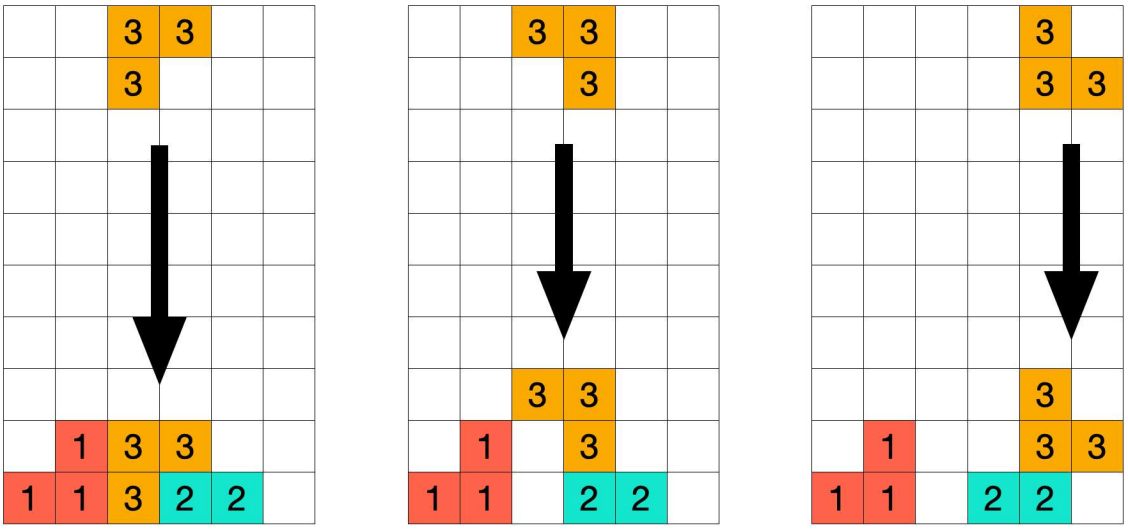
- 对于 50% 的数据，隐藏表达式至多只包含两个运算符，且运算符只有加法和减法。
- 对于 100% 的数据，$1 \leq n \leq 999,999$。保证至少存在一个计算结果等于 n 的表达式。

第六题 (p6.cpp, 20 分): 俄罗斯方块

“俄罗斯方块”是一款经典的电子游戏。你现在希望编写一个可以自动玩“俄罗斯方块”的人工智能。“俄罗斯方块”初始时有一个高度无限、宽度是 6 的矩形，并且矩形中所有的小方格都是空的。你将按顺序得到一系列零件，每个零件是以下两种类型之一，分别命名为 A 和 B：



在得到一个零件后，你可以先将零件任意旋转，然后使零件从足够高处垂直下落到矩形的任意列中（但零件不能超出矩形的左右边界）。零件在垂直下落的过程中，在即将触碰到矩形底部或任意一个之前已经被固定住的零件后会被立即固定。如下图所示，编号为 1（类型为 A）和 2（类型为 B）的零件已被固定，此时固定第三个零件的若干可能方法有：



现在你已经得到了所有零件给出的顺序，你的任务是按照顺序尽可能紧凑、没有空隙地固定所有的零件，以得到尽可能高的分数。具体来说，你给出的固定方案中不能包含超过 10 个未被填满的非空行。

输入格式

输入一行一个仅由大写字母“A”和“B”字符组成的字符串，代表你得到的零件序列。

输出格式

按照以下规则将你固定的所有零件绘制出来（参考样例输出）：

- 从最高的非空行开始，从高到低输出矩形中的每一行；
- 每一行的输出都包含 7 个竖线“|”，用于分隔 6 个方格中固定的零件编号。如果一个方格中没有固定任何零件，则输出不超过 10 个空格；
- 必须保证固定方案满足游戏规则，且未被填满的非空行不超过 10 个。

你可以在数字和竖线之间输出不超过 10 个空格使你的方案更容易阅读（样例输出中包含了额外的空格）。评测时这些空格将被忽略。

样例输入 1

```
AABAABAAABAAABB
```

样例输出 1

```
| 13 |  13 |  14 |  14 |     |     |
| 13 |   9 |  11 |  11 |  15 |  15 |
|  9 |   9 |  10 |  11 |  12 |  12 |
|  6 |   6 |  10 |   7 |  12 |   8 |
|  5 |   5 |   7 |   7 |   8 |   8 |
|  5 |   1 |   2 |   4 |   4 |   3 |
|  1 |   1 |   2 |   2 |   4 |   3 |
```

样例输入 2

```
AAAAABBBBBAAABABABABAABABAABBAAB
```

样例输出 2

```
|29|  |30|  |31|32|
|29|30|30|31|31|32|
|25|26|26|27|28|28|
|25|21|26|27|27|24|
|21|21|22|22|24|24|
|18|18|22|19|23|23|
|17|17|19|19|20|20|
|17|14|15|15|16|16|
|10|14|11|15|13|13|
|10|11|11|12|12|13|
| 7| 7| 8| 8|12| 9|
| 2| 2| 4| 4| 6| 9|
| 2| 1| 4| 3| 6| 5|
```

```
| 1| 1| 3| 3| 5| 5|
```

说明：数字和竖线之间可以有不超过 10 个空格。

数据规模

- 对于 50% 的数据，输入的字符串可以写成是某个长度不超过 3 的字符串的多次重复，例如“ABABABAB...AB”或“BBABBABBA...BBA”；
- 对于 100% 的数据，零件的数量（即输入字符串的长度）不超过 $10,000$。