2020 年 CCF 非专业级软件能力认证
入门级第二轮
2020 CCF CSP-J2

时间：2020 年 11 月 7 日 08:30 ～ 12:00

题目名称	优秀的拆分	直播获奖	表达式	方格取数
题目类型	传统型	传统型	传统型	传统型
目录	power	live	expr	number
可执行文件名	power	live	expr	number
输入文件名	power.in	live.in	expr.in	number.in
输出文件名	power.out	live.out	expr.out	number.out
时间限制	1.0 秒	1.0 秒	1.0 秒	1.0 秒
内存限制	256 MB	256 MB	256 MB	256 MB
测试点数目	20	20	20	20

提交源程序文件名

C++ 语言	power.cpp	live.cpp	expr.cpp	number.cpp
C 语言	power.c	live.c	expr.c	number.c
Pascal 语言	power.pas	live.pas	expr.pas	number.pas

编译选项

C++ 语言	-lm
C 语言	-lm
Pascal 语言	

注意事项（请选手仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/C++ 中函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
3. 提交的程序代码文件的放置位置请参照各省的具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
6. 程序可使用的栈内存空间限制与题目的内存限制一致。
7. 全国统一评测时采用的机器配置为：Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz，内存 32GB。上述时限以此配置为准。
8. 只提供 Linux 格式附加样例文件。
9. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以其为准。

优秀的拆分（power）

【题目描述】

一般来说，一个正整数可以拆分成若干个正整数的和。例如，$1=1$，$10=1+2+3+4$ 等。

对于正整数 n 的一种特定拆分，我们称它为“优秀的”，当且仅当在这种拆分下，n 被分解为了若干个不同的 2 的正整数次幂。注意，一个数 x 能被表示成 2 的正整数次幂，当且仅当 x 能通过正整数个 2 相乘在一起得到。

例如，$10=8+2=2^3+2^1$ 是一个优秀的拆分。但是，$7=4+2+1=2^2+2^1+2^0$ 就不是一个优秀的拆分，因为 1 不是 2 的正整数次幂。

现在，给定正整数 n，你需要判断这个数的所有拆分中，是否存在优秀的拆分。若存在，请你给出具体的拆分方案。

【输入格式】

输入文件名为 power.in。

输入文件只有一行，一个正整数 n，代表需要判断的数。

【输出格式】

输出文件名为 power.out。

如果这个数的所有拆分中，存在优秀的拆分。那么，你需要从大到小输出这个拆分中的每一个数，相邻两个数之间用一个空格隔开。可以证明，在规定了拆分数字的顺序后，该拆分方案是唯一的。

若不存在优秀的拆分，输出“-1”（不包含双引号）。

【样例 1 输入】

6

【样例 1 输出】

4 2

【样例 1 解释】

$6=4+2=2^2+2^1$ 是一个优秀的拆分。注意，$6=2+2+2$ 不是一个优秀的拆分，因为拆分成的 3 个数不满足每个数互不相同。

【样例 2 输入】

7

【样例 2 输出】

-1

【样例 3】

见选手目录下的 power/power3.in 与 power/power3.ans。

【数据范围与提示】

对于 20% 的数据，$n \leq 10$。
对于另外 20% 的数据，保证 n 为奇数。
对于另外 20% 的数据，保证 n 为 2 的正整数次幂。
对于 80% 的数据，$n \leq 1024$。
对于 100% 的数据，$1 \leq n \leq 1 \times 10^7$。

直播获奖（live）

【题目描述】

NOI2130 即将举行。为了增加观赏性，CCF 决定逐一评出每个选手的成绩，并直播即时的获奖分数线。本次竞赛的获奖率为 $w\%$，即当前排名前 $w\%$ 的选手的最低成绩就是即时的分数线。

更具体地，若当前已评出了 p 个选手的成绩，则当前计划获奖人数为 $\max(1, \lfloor p \times w\% \rfloor)$，其中 w 是获奖百分比，$\lfloor x \rfloor$ 表示对 x 向下取整，$\max(x, y)$ 表示 x 和 y 中较大的数。如有选手成绩相同，则所有成绩并列的选手都能获奖，因此实际获奖人数可能比计划中多。

作为评测组的技术人员，请你帮 CCF 写一个直播程序。

【输入格式】

输入文件名为 live.in。

第 1 行两个正整数 n, w。分别代表选手总数与获奖率。

第 2 行有 n 个非负整数，依次代表逐一评出的选手成绩。

【输出格式】

输出文件名为 live.out。

只有一行，包含 n 个非负整数，依次代表选手成绩逐一评出后，即时的获奖分数线。相邻两个整数间用一个空格分隔。

【样例 1 输入】

```
10 60
200 300 400 500 600 600 0 300 200 100
```

【样例 1 输出】

```
200 300 400 400 400 500 400 400 300 300
```

【样例 1 解释】

已评测选手人数	1	2	3	4	5	6	7	8	9	10
计划获奖人数	1	1	1	2	3	3	4	4	5	6

已评测选手的分数从高到低排列（其中，分数线用**粗体**标出）	**200**	**300** 200	**400** 300 200	500 **400** 300 200	600 500 **400** 300 200	600 600 **500** 400 300 200	600 600 500 **400** 300 200 0	600 600 500 **400** 300 300 200 0	600 600 500 400 **300** 300 200 200 0	600 600 500 400 300 **300** 200 200 100 0

注意，在第 9 名选手的成绩评出之后，计划获奖人数为 5 人，但由于有并列，因此实际会有 6 人获奖。

【样例 2 输入】

```
10 30
100 100 600 100 100 100 100 100 100 100
```

【样例 2 输出】

```
100 100 600 600 600 600 100 100 100 100
```

【样例 3】

见选手目录下的 live/live3.in 与 live/live3.ans。

【数据范围与提示】

测试点编号	n
1~3	= 10
4~6	= 500
7~10	= 2000
11~17	= 10000
18~20	= 100000

对于所有测试点，每个选手的成绩均为不超过 600 的非负整数，获奖百分比 w 是一个正整数且 $1 \le w \le 99$。

在计算计划获奖人数时，如用浮点类型的变量（如 C/C++中的 float、double，Pascal 中的 real、double、extended 等）存储获奖比例 $w\%$，则计算 $5 \times 60\%$ 时的结果可能为 3.000001，也可能为 2.999999，向下取整后的结果不确定。因此，建议仅使用整型变量，以计算出准确值。

表达式（expr）

【题目描述】

小 C 热衷于学习数理逻辑。有一天，他发现了一种特别的逻辑表达式。在这种逻辑表达式中，所有操作数都是变量，且它们的取值只能为 0 或 1，运算从左往右进行。如果表达式中有括号，则先计算括号内的子表达式的值。特别的，这种表达式有且仅有以下几种运算：

1. 与运算：$a \& b$。当且仅当 a 和 b 的值都为 1 时，该表达式的值为 1。其余情况该表达式的值为 0。
2. 或运算：$a \mid b$。当且仅当 a 和 b 的值都为 0 时，该表达式的值为 0。其余情况该表达式的值为 1。
3. 取反运算：$!a$。当且仅当 a 的值为 0 时，该表达式的值为 1。其余情况该表达式的值为 0。

小 C 想知道，给定一个逻辑表达式和其中每一个操作数的初始取值后，再取反某一个操作数的值时，原表达式的值为多少。

为了化简对表达式的处理，我们有如下约定：

表达式将采用**后缀表达式**的方式输入。后缀表达式的定义如下：

1. 如果 E 是一个操作数，则 E 的后缀表达式是它本身。
2. 如果 E 是 $E_1\ op\ E_2$ 形式的表达式，其中 op 是任何二元操作符，且优先级不高于 E_1、E_2 中括号外的操作符，则 E 的后缀式为 $E_1'\ E_2'\ op$，其中 E_1'、E_2' 分别为 $E1$、$E2$ 的后缀式。
3. 如果 E 是 (E_1) 形式的表达式，则 E_1 的后缀式就是 E 的后缀式。

同时为了方便，输入中：

a) 与运算符（&）、或运算符（|）、取反运算符（!）的左右**均有一个空格**，但表达式末尾**没有空格**。

b) 操作数由小写字母 x 与一个正整数拼接而成，正整数表示这个变量的下标。例如：x10，表示下标为 10 的变量 x_{10}。数据保证**每个变量在表达式中出现恰好一次**。

【输入格式】

输入文件名为 expr.in。

第一行包含一个字符串 s，表示上文描述的表达式。

第二行包含一个正整数 n，表示表达式中变量的数量。表达式中变量的下标为 $1,2,\ldots,n$。

第三行包含 n 个整数，第 i 个整数表示变量 x_i 的初值。

第四行包含一个正整数 q，表示询问的个数。

接下来 q 行，每行一个正整数，表示需要取反的变量的下标。注意，每一个询问的修改都是**临时的**，即之前询问中的修改不会对后续的询问造成影响。

数据保证输入的表达式合法。变量的初值为 0 或 1。

【输出格式】

输出文件名为 expr.out。

输出一共有 q 行，每行一个 0 或 1，表示该询问下表达式的值。

【样例 1 输入】

```
x1 x2 & x3 |
3
1 0 1
3
1
2
3
```

【样例 1 输出】

```
1
1
0
```

【样例 1 解释】

该后缀表达式的中缀表达式形式为 $(x_1 \& x_2) \mid x_3$。

对于第一次询问，将 x_1 的值取反。此时，三个操作数对应的赋值依次为 0，0，1。原表达式的值为 $(0 \& 0) \mid 1 = 1$。

对于第二次询问，将 x_2 的值取反。此时，三个操作数对应的赋值依次为 1，1，1。原表达式的值为 $(1 \& 1) \mid 1 = 1$。

对于第三次询问，将 x_3 的值取反。此时，三个操作数对应的赋值依次为 1，0，0。原表达式的值为 $(1 \& 0) \mid 0 = 0$。

【样例 2 输入】

```
x1 ! x2 x4 | x3 x5 ! & & ! &
5
0 1 0 1 1
3
1
3
5
```

【样例 2 输出】

0
1
1

【样例 2 解释】

该表达式的中缀表达式形式为 $(!x_1)\&(!((x_2|x_4)\&(x_3\&(!x_5))))$。

【样例 3】

见选手目录下的 expr/expr3.in 与 expr/expr3.ans。

【数据范围与提示】

对于 20% 的数据，表达式中有且仅有与运算（&）或者或运算（|）。

对于另外 30% 的数据，$|s| \leq 1000$，$q \leq 1000$，$n \leq 1000$。

对于另外 20% 的数据，变量的初值全为 0 或全为 1。

对于 100% 的数据，$1 \leq |s| \leq 1 \times 10^6$，$1 \leq q \leq 1 \times 10^5$，$2 \leq n \leq 1 \times 10^5$。

其中，$|s|$ 表示字符串 s 的长度。

方格取数（number）

【题目描述】

设有 $n \times m$ 的方格图，每个方格中都有一个整数。现有一只小熊，想从图的左上角走到右下角，每一步只能向上、向下或向右走一格，并且不能重复经过已经走过的方格，也不能走出边界。小熊会取走所有经过的方格中的整数，求它能取到的整数之和的最大值。

【输入格式】

输入文件名为 number.in。
第 1 行两个正整数 n, m。
接下来 n 行每行 m 个整数，依次代表每个方格中的整数。

【输出格式】

输入文件名为 number.out。
一个整数，表示小熊能取到的整数之和的最大值。

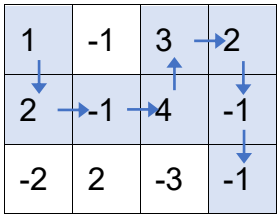
【样例 1 输入】

```
3 4
1 -1 3 2
2 -1 4 -1
-2 2 -3 -1
```

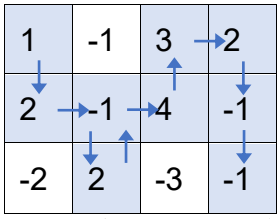
【样例 1 输出】

```
9
```

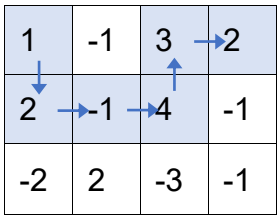
【样例 1 解释】



按上述走法，取到的数之和为 1 + 2 + (-1) + 4 + 3 + 2 + (-1) + (-1) = 9，可以证明为最大值。



注意，上述走法是错误的，因为第 2 行第 2 列的方格走过了两次，而根据题意，不能重复经过已经走过的方格。



另外，上述走法也是错误的，因为没有走到右下角的终点。

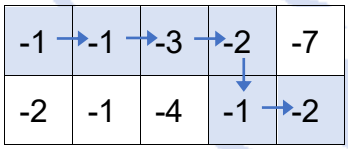
【样例 2 输入】

```
2 5
-1 -1 -3 -2 -7
-2 -1 -4 -1 -2
```

【样例 2 输出】

```
-10
```

【样例 2 解释】



按上述走法，取到的数之和为(-1) + (-1) + (-3) + (-2) + (-1) + (-2) = -10，可以证明为最大值。因此，请注意，取到的数之和的最大值也可能是负数。

【样例 3】

见选手目录下的 number/number3.in 与 number/number3.ans。

【数据范围与提示】

对于 20% 的数据，$n, m \leq 5$。
对于 40% 的数据，$n, m \leq 50$。
对于 70% 的数据，$n, m \leq 300$。
对于 100% 的数据，$1 \leq n, m \leq 1000$。方格中整数的绝对值不超过 10^4。