1. 下一个更大元素

(nextGreaterElements.cpp)

限制：1 秒　　256MB

题目描述：给定一个循环数组 nums （ nums 最后一个元素的下一个元素是 nums[0] ），返回 nums 中每个元素的下一个更大元素 。

数字 x 的下一个更大的元素是按数组遍历顺序，这个数字之后的第一个比它更大的数，这意味着你应该循环地搜索它的下一个更大的数。如果不存在，则输出 -1 。

输入：(nextGreaterElements.in)

输入一个数组。

输出：(nextGreaterElements.out)

输出 nums 的下一个更大元素集合，每个元素用一个空格相隔。

输入样例 1：

1　2　1

输出样例 1：

2　-1　2

样例 1 解释：

第一个 1 的下一个更大的数是 2；
数字 2 找不到下一个更大的数；
第二个 1 的下一个最大的数需要循环搜索，结果也是 2。

输入样例 2：

1　2　3　4　3

输出样例 2：

2　3　4　-1　4

数据范围限制：

nums 的长度不超过 10^4。
-10^9 <= nums[i] <= 10^9

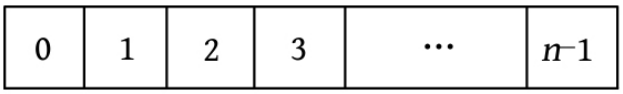
2. 区块世界

(block.cpp/.c)

限制：1S　256MB

题目描述：

在早期的人工智能规划和机器人研究中使用了一个区块世界，在这个世界中，机器人手臂执行涉及区块操作的任务。问题是要解析一系列命令，这些命令指导机器人手臂如何操作平板上的块。最初，有 n 个区块（编号为 0～n−1），对于所有 0≤i<n−1 的情况，区块 b_i 与区块 b_{i+1} 相邻，如下图所示。



用于操纵块的有效命令如下。

- move a onto b：把 a 和 b 上方的块全部放回初始位置，然后把 a 放到 b 上方。
- move a over b：把 a 上方的块全部放回初始位置，然后把 a 放到 b 所在块堆的最上方。
- pile a onto b：把 b 上方的块全部放回初始位置，然后把 a 和 a 上方所有的块整体放到 b 上方。
- pile a over b：把 a 和 a 上方所有的块整体放到 b 所在块堆的最上方。
- quit：结束标志。

任何 a=b 或 a 和 b 在同一块堆中的命令都是非法命令。所有非法命令都应被忽略。

输入：(block.in)

输入的第 1 行为整数 n（0<n<25），表示区块世界中的块数。后面是一系列块命令，每行一个命令。在遇到 quit 命令之前，程序应该处理所有命令。所有命令都将采用上面指定的格式，不会有语法错误的命令。

输出：(block.out)

输出应该包含区块世界的最终状态。每一个区块 i（0≤i<n）后面都有一个冒号。如果上面至少有一个块，则冒号后面必须跟一个空格，后面跟一个显示在该位置的块列表，每个块号与其他块号之间用空格隔开。不要在行末加空格。

输入样例	输出样例
10 move 9 onto 1 move 8 over 1 move 7 over 1 move 6 over 1 pile 8 over 6 pile 8 over 5 move 2 over 1 move 4 over 9 quit	0: 0 1: 1 9 2 4 2: 3: 3 4: 5: 5 8 7 6 6: 7: 8: 9:

3. City Game

Time Limit: 1Sec Memory Limit: 256 MB
(city.cpp/c/pas)

Description

这片城市土地被分成 N*M 个格子，每个格子里写着'R'或者'F'，R 代表这块土地被赐予了 rainbow，F 代表这块土地被赐予了 freda。
现在 freda 要在这里卖萌。。。它要找一块矩形土地，要求这片土地都标着'F'并且面积最大。
但是 rainbow 和 freda 的 OI 水平都弱爆了，找不出这块土地，而蓝兔也想看 freda 卖萌（她显然是不会编程的……），所以它们决定，如果你找到的土地面积为 S，它们每人给你 3*S 两银子。

Input

第一行两个整数 N,M，表示矩形土地有 N 行 M 列。
接下来 N 行，每行 M 个用空格隔开的字符'F'或'R'，描述了矩形土地。

Output

输出一个整数，表示你能得到多少银子，即(3*最大'F'矩形土地面积)的值。

Sample Input

```
5 6
R F F F F F
F F F F F F
R R R F F F
F F F F F F
F F F F F F
```

Sample Output

```
45
```

HINT

对于 50%的数据，1<=N,M<=200

对于 100%的数据，1<=N,M<=1000

4. 移动盒子

(move.cpp/.c)

限制：1S　256MB

题目描述：

一行有 n 个盒子，从左到右编号为 1～n。模拟以下 4 种命令。

- 1 X Y：将盒子 X 移动到 Y 的左侧（如果 X 已经在 Y 的左侧，则忽略此项）。
- 2 X Y：将盒子 X 移动到 Y 的右侧（如果 X 已经在 Y 的右侧，则忽略此项）。
- 3 X Y：交换盒子 X 和 Y 的位置。
- 4：翻转整行盒子序列。

以上命令保证有效，即 X 不等于 Y。

举例说明：有 6 个盒子，执行 1 1 4，即 1 移动到 4 的左侧，变成 2 3 1 4 5 6。然后执行 2 3 5，即 3 移动到 5 的右侧，变成 2 1 4 5 3 6。接着执行 3 1 6，即交换 1 和 6 的位置，变成 2 6 4 5 3 1。最后执行 4，即翻转整行序列，变成 1 3 5 4 6 2。

输入：(move.in)

最多有 10 个测试用例。每个测试用例的第 1 行都包含两个整数 n 和 m（1≤n, m≤100 000），下面的 m 行，每行都包含一个命令。

输出：(move.out)

对于每个测试用例，都单行输出奇数索引位置的数字总和。

输入样例	输出样例
6 4 1 1 4 2 3 5 3 1 6 4 6 3 1 1 4 2 3 5 3 1 6 100000 1 4	Case 1: 12 Case 2: 9 Case 3: 2500050000

5. 表达式计算

Time Limit: 1Sec Memory Limit: 256 MB
(expression.cpp/c/pas)

Description

给出一个表达式,其中运算符仅包含+,-,*,/,^（加 减 乘 整除 乘方），要求求出表达式的最终值。数据可能会出现括号情况，还有可能出现多余括号情况。数据保证不会出现>=2^31 的答案，数据可能会出现负数情况。

Input

仅一行，即为表达式。

Output

仅一行，表达式算出的结果。

Sample Input

(2+2)^(1+1)

Sample Output

16

HINT

表达式总长度<=30。