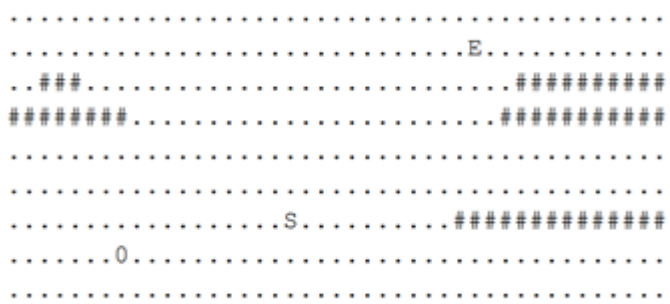
字节跳动 2018 校招算法方向（第三批）

1. [编程题]编程题 1

时间限制：C/C++ 1 秒，其他语言 2 秒
空间限制：C/C++ 64M，其他语言 128M

有一个推箱子的游戏，一开始的情况如下图:



上图中，'.' 表示可到达的位置，'#' 表示不可到达的位置，其中 S 表示你起始的位置，0 表示初始箱子的位置，E 表示预期箱子的位置，你可以走到箱子的上下左右任意一侧，将箱子向另一侧推动。如下图将箱子向右推动一格;
..S0.. -> ...S0.

注意不能将箱子推动到'#'上，也不能将箱子推出边界;

现在，给你游戏的初始样子，你需要输出最少几步能够完成游戏，如果不能完成，则输出-1。

输入描述:
第一行为 2 个数字,n, m，表示游戏盘面大小有 n 行 m 列(5< n, m < 50);
后面为 n 行字符串,每行字符串有 m 字符，表示游戏盘面;

输出描述:
一个数字,表示最少几步能完成游戏,如果不能,输出-1;

输入例子 1:
3 6
.S#..E
.#.0..
.....

输出例子 1:

11

2. [编程题]编程题 2

时间限制：C/C++ 1 秒，其他语言 2 秒
空间限制：C/C++ 64M，其他语言 128M

有 n 个房间，现在 i 号房间里的人需要被重新分配，分配的规则是这样的：先让 i 号房间里的人全都出来，接下来按照 i+1, i+2, i+3, ... 的顺序依此往这些房间里放一个人，n 号房间的的下一个房间是 1 号房间，直到所有的人都被重新分配。

现在告诉你分配完后每个房间的人数以及最后一个人被分配的房间号 x，你需要求出分配前每个房间的人数。数据保证一定有解，若有多解输出任意一个解。

输入描述:
第一行两个整数 n, x (2<=n<=10^5, 1<=x<=n)，代表房间房间数量以及最后一个人被分配的房间号；
第二行 n 个整数 a_i(0<=a_i<=10^9) ，代表每个房间分配后的人数。

输出描述:
输出 n 个整数，代表每个房间分配前的人数。

输入例子 1:
3 1
6 5 1

输出例子 1:
4　4 4

3. [问答题]

题目描述
在生产环境，我们常常要存储一些像服务参数、功能开关之类的键值。传统的做法是把配置都写到文件里，然后同步到线上每台机器上。随着机器变多，配置文件变得难以管理，并且容易出现不一致的情况。我们希望设计一个配置服务来解决这个问题。

统一配置服务可能会存在以下问题：由于是非常核心的服务，如果存在单节点问题对服务可用性影响非常大；线上可能读取非常频繁，尽可能提供高性能的服务同时，也要考虑横向扩容能力；需要保证配置在期望的时间内下发与更新；

请设计一个存储服务，包含但不限于以下角色：服务端（可能由多个节点组成），客户端（读取、写入一个配置），其他（如旁路的监控等）；

系统假设：

1、存储量都在 1GB 以内，单机内存可以存储下；

2、每秒写入在 1000 以内

3、每秒读取在 1000000 以上

4、使用尽量少的节点

5、无论什么时候，服务总是可以读写

6、允许故障期间读到老的配置数据

7、故障恢复后，数据保持同步

4. [问答题]

题目描述
以下函数用于找到整数矩阵 matrix 中，元素之和最大的 n 行 m 列的子矩阵的元素之和。请指出程序代码中错误的地方（问题不止一处，请尽量找出所有你认为错误的地方），并在不新增代码行的情况下将问题修复。

```
1 int maxSubmatrixSum(std::vector<std::vector<int>> matrix,
2                      int n, int m) {
3     int base_sum;
4     for (int i = 0; i < n; i++){
5         for (int j = 0; j < m; j++){
6             base_sum += matrix[i][j];
7         }
8     }
9     int result = 0;
10    for (int i = 0; i + n < matrix.size(); i++) {
11        if(i    > 0){
12            for (int y = 0; y < m; y++){
13                base_sum += matrix[i + n][y] - matrix[i - 1][y];
14            }
15        }
16        int real_sum = base_sum;
17        if (real_sum    > result) {
```

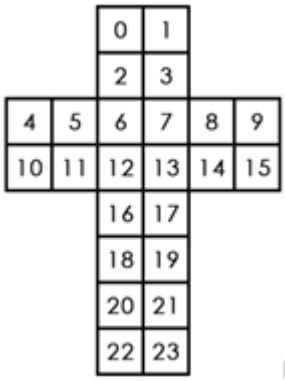
```
            result = real_sum;
        }
        for (int j = 0; j + m < matrix.size(); j++) {
            for (int x = 0; x < n; x++) {
                real_sum += matrix[x][j + m] - matrix[x][j - 1];
            }
            if (real_sum > result) {
                result = real_sum;
            }
        }
    }
    return result;
```

5. [编程题]附加题

时间限制：C/C++ 1 秒，其他语言 2 秒
空间限制：C/C++ 32M，其他语言 64M

二阶魔方又叫小魔方，是 2*2*2 的立方形结构。每一面都有 4 个块，共有 24 个块。每次操作可以将任意一面逆时针或者顺时针旋转 90°，如将上面逆时针旋转 90°操作如下。

Nero 在小魔方上做了一些改动，用数字替换每个块上面的颜色，称之为数字魔方。魔方上每一面的优美度就是这个面上 4 个数字的乘积，而魔方的总优美度就是 6 个面优美度总和。现在 Nero 有一个数字魔方，他想知道这个魔方在操作不超过 5 次的前提下能达到的最大优美度是多少。
魔方展开后每一块的序号如下图：



输入描述:
输入一行包含 24 个数字，按序号顺序给出魔方每一块上面的数字。所有数大小范围为[-100,100]。

输出描述:
输出一行包含一个数字，表示最大优美度。

输入例子 1:
2 -3 -2 3 7 -6 -6 -7 9 -5 -9 -3 -2 1 4 -9 -1 -10 -5 -5 -10 -4 8 2

输出例子 1:
8281