问题 A: 序列

给定一个整数序列 a1,a2,……,an ， 序列中每一个元素都是 1,2,3 中的任意一个。你必须从其中替换尽可能少的元素，以使序列中的所有数字都相等。

输入格式

第一行输入一个整数 n ($1 \leqslant n \leqslant 10^6$). 第二行输入 n 哥整数 a1,a2,...,an ($1 \leqslant ai \leqslant 3$).

输出格式

输出一个最小值。

Examples

输入	输出
9 1 3 2 2 2 1 1 2 3	5

问题 B: 软件公司

zz 的软件公司的名字由 N 小写的英文字母组成，随着公司规模的扩大，zz 决定重新设计公司的名字。
为此，公司先后聘请了 M 位设计人员，每位设计师都对名字的修改做出了贡献。第 i 个设计师的工作是将最新版本的所有的字母 xi 变为 yi,yi 变为 xi

在最后一个设计师的工作之后，名字的版本变成了公司的新名称。

输入格式

输入的第一行包含两个整数 n 和 m (1<=n，m<=200000） — 初始名称的长度和雇用的设计师数量。第二行包括 n 个小写英文字母，代表公司的原始名称。
下一个 m 行包含每位设计师操作的描述：第 i 行包含两个空格分隔的小写英文字母 xi 和 yi

输出格式

打印公司的新名称。

Examples

Input

6 1

police

p m

Output

molice

Input

11 6

abacabadaba

a b

b c

a d

e g

f a

b b

Output

cdcbcdcfcdc

第二个样例的变化过程为:

abacabadaba → babcbabdbabbabcbabdbab → cacbcacdcac
cacbcacdcac → cdcbcdcacdc
cdcbcdcacdc → cdcbcdcacdc
cdcbcdcacdc → cdcbcdcfcdc
cdcbcdcfcdc → cdcbcdcfcdc

问题 C: zz 的考试

zz 想通过 n 次考试来获得奖学金。如果所有考试的平均分数不低于 avg，他就能获得奖学金。考试成绩较差时，zz 也可以通过写论文来提高分数，但提高后的成绩不能超过 r（加完得到的分数不能超过 r）。ai 是 zz 第 i 次考试已经获得的分数，为了将第 i 次考试的成绩提高 1 分，zz 必须写 bi 篇论文。通过写论文他可以多次提高考试成绩。

为了获得奖学金，zz 最少需要写多少篇论文？

输入格式
第一行包含三个整数 n，r，avg（1≤n≤10^5，1≤r≤10^9，1≤avg≤min(r,10^6)）--分别是考试次数，最高成绩和要求的平均分。

下面的 n 行中的每一行都包含空间分隔的整数 ai 和 bi（1≤ai≤r，1≤bi≤10^6）。
输出格式
在第一行打印最小的论文数量。
Examples

5 5 4 5 2 4 7 3 1 3 2 2 5	4
2 5 4 5 2 5 2	0

Input
Note
在第一个样例中，Vanya 可以在第三次考试中写 2 篇论文，使他的成绩提高 2 分，在第四次考试中写 2 篇论文，使他的成绩提高 1 分。
在第二个例子中，Vanya 不需要写任何论文，因为他的总平均分已经超过了平均水平。

输入样例 复制
5 5 4
5 2
4 7
3 1
3 2
2 5
输出样例 复制
4

问题 D: 美国奥林匹克竞赛

题目描述

凯文刚刚以长度为 n 的二进制字符串的形式公布了他在美国奥林匹克竞赛（USACO）中令人失望的成绩。凯文字符串中的每个字符代表凯文在奥林匹克竞赛 n 个问题中的一个问题上的得分——正确为“1”，否则为“0”。

然而，他认为他的得数不应该是各题分数的总和，而应该是他的字符串中最长的交替子序列的长度。这里，我们将字符串的交替子序列定义为不一定连续的子序列，其中没有两个连续元素相等。例如{0,1, 0, 1}, {1, 0, 1} 和{1,0, 1, 0}是交替序列， {1,0, 0}和{0,1,0, 1, 1} 不是。

凯文是一个狡猾的小混混，他愿意侵入 USACO 数据库来提高自己的分数。为了变得微妙，他决定只翻转一个子字符串，即取分数的一个连续的非空子字符串，并将该子字符串中的所有“0”更改为“1”，反之亦然。在这样的操作之后，凯文想知道他的字符串可能具有的最长交替子序列的长度。

输入格式

第一行包含关于奥运会的问题数量 n（1≤n≤100000）。

以下行包含长度为 n 的二进制字符串，表示凯文在 USACO 上的结果。

输出格式

输出单个整数，即 Kevin 在翻转单个子字符串后可以在字符串中创建的最长交替子序列的长度。

样例

Input

8

10000011

Output

5

Input

2

01

Output

2

分析

在第一个示例中，Kevin 可以把串 10000011 中加粗的部分翻转, 使其成为 10011011, 这样这个串中最长的 01 间隔子串为 10011011 , 长度为 5.

在第二个示例中，凯文可以翻转整个字符串，但仍然有相同的分数。

输入样例 复制

8

10000011

输出样例

5

问题 E：消消乐

由于手上（更确实的，蹄子上）有大把的空余时间，Farmer John 的农场里的奶牛经常玩电子游戏消磨时光。

她们最爱的游戏之一是基于一款人类中流行的电子游戏噗呦噗呦的奶牛版；名称当然叫做哞呦哞呦。

哞呦哞呦是在一块细长的棋盘上进行的游戏，高 N 格，宽 10 格。

这是一个 N=6 的棋盘的例子：

0000000000
0000000300
0054000300
1054502230
2211122220
1111111223

每个格子中或者是空的（用 0 表示），或者是九种颜色之一的干草捆（用字符 1..9 表示）。

重力会使得干草捆下落，所以没有干草捆的下方是 0。

如果两个格子水平或垂直方向直接相邻，并且为同一种非 0 颜色，那么这两个格子就属于同一个连通区域。

任意时刻出现至少 K 个格子构成的连通区域，其中的干草捆就会全部消失，变为 0。

如果同时出现多个这样的连通区域，它们同时消失。

随后，重力可能会导致干草捆向下落入某个变为 0 的格子。

由此形成的新的布局中，又可能出现至少 K 个格子构成的连通区域。

若如此，它们同样也会消失（如果又有多个这样的区域，则同时消失），然后重力又会使得剩下的方块下落，这一过程持续进行，直到不存在大小至少为 K 的连通区域为止。

给定一块哞呦哞呦棋盘的状态，输出这些过程发生之后最终的棋盘的图案。

输入格式

输入的第一行包含 N 和 K。

以下 N 行给出了棋盘的初始状态。

输出格式

输出 N 行，描述最终的棋盘状态。

1≤N≤100, 1≤K≤10N

6 3 0000000000 0000000300 0054000300 1054502230 2211122220 1111111223	0000000000 0000000000 0000000000 0000000000 1054000000 2254500000

```cpp
#include <iostream>
using namespace std;
unsigned short f(unsigned short x) {
    x ^= x << 6;
    x ^= x >> 8;
    return x;
}
int main() {
    unsigned short x;
    cin >> x;
    unsigned short y = f(x);
    cout << y << endl;
    return 0;
}
```

假设输入的 x 是不超过 65535 的自然数，完成下面的判断题和单选题:

判断题

1. 当输入非零时，输出一定不为零。 ()
2. （2 分）将 `f` 函数的输入参数的类型改为 `unsigned int`，程序的输出不变。 ()
3. 当输入为 `65535` 时，输出为 `63`。 ()
4. 当输入为 `1` 时，输出为 `64`。 ()

单选题

5. 当输入为 `512` 时，输出为 ()。
6. 当输入为 `64` 时，执行完第 5 行后 `x` 的值为 ()。

1. ○ A. 正确
   ○ B. 错误
2. ○ A. 正确
   ○ B. 错误
3. ○ A. 正确
   ○ B. 错误
4. ○ A. 正确
   ○ B. 错误
5. ○ A. `33280`
   ○ B. `33410`
   ○ C. `33106`
   ○ D. `33346`
6. ○ A. `8256`
   ○ B. `4130`
   ○ C. `4128`
   ○ D. `4160`

```cpp
#include <iostream>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;

long long solve1(int n) {
    vector<bool> p(n + 1, true);
    vector<long long> f(n + 1, 0), g(n + 1, 0);
    f[1] = 1;
    for (int i = 2; i * i <= n; i++) {
        if (p[i]) {
            vector<int> d;
            for (int k = i; k <= n; k *= i)
                d.push_back(k);
            reverse(d.begin(), d.end());
            for (int k : d) {
                for (int j = k; j <= n; j += k) {
                    if (p[j]) {
                        p[j] = false;
                        f[j] = i;
                        g[j] = k;
                    }
                }
            }
        }
    }
    for (int i = sqrt(n) + 1; i <= n; i++) {
        if (p[i]) {
            f[i] = i;
            g[i] = i;
        }
    }
    long long sum = 1;
    for (int i = 2; i <= n; i++) {
        f[i] = f[i / g[i]] * (g[i] * f[i] - 1) / (f[i] - 1);
        sum += f[i];
    }
    return sum;
}

long long solve2(int n) {
    long long sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i * (n / i);
    }
    return sum;
}

int main() {
    int n;
    cin >> n;
    cout << solve1(n) << endl;
    cout << solve2(n) << endl;
    return 0;
}
```

假设输入的 n 是不超过 1000000 的自然数，完成下面的判断题和单选题:

判断题

1. 将第 15 行删去，输出不变。 ()
2. 当输入为 `10` 时，输出的第一行大于第二行。 ()
3. （2 分） 当输入为 `1000` 时，输出的第一行与第二行相等。 ()

单选题

4. `solve1(n)` 的时间复杂度为（）。
5. `solve2(n)` 的时间复杂度为（）。
6. 当输入为 `5` 时，输出的第二行为（）。

1. A. 正确
   B. 错误
2. A. 正确
   B. 错误
3. A. 正确
   B. 错误
4. A. $O(n \log^2 n)$
   B. $O(n)$
   C. $O(n \log n)$
   D. $O(n \log \log n)$
5. A. $O(n^2)$
   B. $O(n)$
   C. $O(n \log n)$
   D. $O(n\sqrt{n})$
6. A. `20`
   B. `21`
   C. `22`
   D. `23`