```
        int eO = 0, eOhasOne = 0;
        for (int curNum : arr) {
                eO ^= curNum;
        }
        int rightOne = eO & (~eO + 1);
        for (int cur : arr) {
                if ((cur & rightOne) != 0) {
                        eOhasOne ^= cur;
                }
        }
        System.out.println(eOhasOne + " " + (eO ^ eOhasOne));
}
```

在其他数都出现 k 次的数组中找到只出现一次的数

【题目】

给定一个整型数组 arr 和一个大于 1 的整数 k。已知 arr 中只有 1 个数出现了 1 次，其他的数都出现了 k 次，请返回只出现了 1 次的数。

【要求】

时间复杂度为 $O(N)$，额外空间复杂度为 $O(1)$。

【难度】

尉 ★★☆☆

【解答】

以下的例子是两个七进制数的无进位相加，即忽略进位的相加，比如：

七进制数 a：　　6 4 3 2 6 0 1

七进制数 b：　　3 4 5 0 1 1 1

无进位相加结果：2 1 1 2 0 1 2

可以看出，两个七进制的数 a 和 b，在 i 位上无进位相加的结果就是(a(i)+b(i))%7。同理，k 进制的两个数 c 和 d，在 i 位上无进位相加的结果就是(c(i)+d(i))%k。那么，如果 k 个相同的 k 进制数进行无进位相加，相加的结果一定是每一位上都是 0 的 k 进制数。

理解了上述过程之后，解这道题就变得简单了，首先设置一个变量 eO，它是一个 32 位的 k 进制数，且每个位置上都是 0。然后遍历 arr，把遍历到的每一个整数都转换为 k 进

制数，然后与 eO 进行无进位相加。遍历结束时，把 32 位的 k 进制数 eORes 转换为十进制整数，就是我们想要的结果。因为 k 个相同的 k 进制数无进位相加，结果一定是每一位上都是 0 的 k 进制数，所以只出现一次的那个数最终就会剩下来。具体请参看如下代码中的 onceNum 方法。

```
public int onceNum(int[] arr, int k) {
        int[] eO = new int[32];
        for (int i = 0; i != arr.length; i++) {
                setExclusiveOr(eO, arr[i], k);
        }
        int res = getNumFromKSysNum(eO, k);
        return res;
}

public void setExclusiveOr(int[] eO, int value, int k) {
        int[] curKSysNum = getKSysNumFromNum(value, k);
        for (int i = 0; i != eO.length; i++) {
                eO[i] = (eO[i] + curKSysNum[i]) % k;
        }
}

public int[] getKSysNumFromNum(int value, int k) {
        int[] res = new int[32];
        int index = 0;
        while (value != 0) {
                res[index++] = value % k;
                value = value / k;
        }
        return res;
}

public int getNumFromKSysNum(int[] eO, int k) {
        int res = 0;
        for (int i = eO.length - 1; i != -1; i--) {
                res = res * k + eO[i];
        }
        return res;
}
```