第 k 个缺失的正整数

给你一个 **严格升序排列** 的正整数数组 `arr` 和一个整数 `k` 。

请你找到这个数组里第 `k` 个缺失的正整数。

示例 1：

输入： `arr = [2,3,4,7,11], k = 5`

输出： `9`

解释： 缺失的正整数包括 `[1,5,6,8,9,10,12,13,...]` 。第 `5` 个缺失的正整数为 `9` 。

示例 2：

输入： `arr = [1,2,3,4], k = 2`

输出： `6`

解释： 缺失的正整数包括 `[5,6,7,...]` 。第 `2` 个缺失的正整数为 `6` 。

我们可以顺序枚举。

用一个变量 current 表示当前应该出现的数，从 1 开始，每次循环都让该变量递增。用一个指针 ptr 指向数组中没有匹配的第一个元素，每轮循环中将该元素和 current 进行比较，如果相等，则指针后移，否则指针留在原地不动，说明缺失正整数 current。我们用 missCount 变量记录缺失的正整数的个数，每次发现有正整数缺失的时候，该变量自增，并且记录这个缺失的正整数，直到我们找到第 k 个缺失的正整数。

```
/*class Solution {
public:
    int findKthPositive(vector<int>& arr, int k) {
        if(arr.size()==0)
            return 0;
        vector<int> nums;
        //根据起始元素 添加前方缺失数字
        for(int i=1;i<arr[0];i++)
        {
            nums.push_back(i);
        }
        //找到原数组间隔缺失数字 添加到数组中
```

```
        for(int i=1;i<arr.size();i++)
        {
            if(arr[i]-arr[i-1]>1)
            {
                int tmp=arr[i]-arr[i-1];
                for(int j=1;j<tmp;j++)
                {
                    nums.push_back(arr[i]+j);
                }
            }
        }
        if(k>nums.size())
        {
            return arr[arr.size()-1]+k;
        }
        else
        {
            return nums[k-1];
        }
    }
};*/

class Solution {
public:
    int findKthPositive(vector<int>& arr, int k) {
        int missCount = 0, lastMiss = -1, current = 1, ptr = 0;
        for (missCount = 0; missCount < k; ++current) {
            if (current == arr[ptr]) {
                ptr = (ptr + 1 < arr.size()) ? ptr + 1 : ptr;
            } else {
                ++missCount;
                lastMiss = current;
            }
        }
        return lastMiss;
    }
};
```