题目一：【26012】--- 字符集合

【题目链接】：

> https://www.nowcoder.com/practice/784efd40ed8e465a84821c8f3970b7b5?tpId=49&&tqId=29297&rp=1&ru=/activity/oj&qru=/ta/2016test/question-ranking

【题目解析】

> 按照顺序输出字符，重复的字符只输出第一次出现的，即后面重复出现的字符不输出

【解题思路】

> 用一个256数组，对已经输出的字符进行标记，在输出每个字符时，在计数数组中检测下该字符是否数据，如果已经输出，则不输出。

```
#include <iostream>
using namespace std;
#include <string>

int main()
{
    string str;
    while(cin>>str)
    {
        int count[256] = {0};
        size_t size = str.size();
        for(size_t i = 0; i < size; ++i)
        {
            // 如果该字符未输出过，则输出
            if(0 == count[str[i]])
                cout<<str[i];

            // 对该字符进行标记，下次遇到则不再输出
            count[str[i]]++;
        }
        cout<<endl;
    }

    return 0;
}
```


题目二：【24984】---元素查找(二分查找变形)

【题目链接】：

> https://www.nowcoder.com/practice/72ff6503455c4a008675e79247ef2a3a?tpId=8&&tqId=11047&rp=1&ru=/activity/oj&qru=/ta/cracking-the-coding-interview/question-ranking

【题目解析】

> 给定一个有序但经过多次移位的数组，移动次数未知，在O(logN)时间复杂度只能找出元素x在数组中的位置。

【解题思路】

> 该题目实际是二分查找的变形，在二分查找的基础上，稍作改变即可，具体方法如下：

> 在解本次前一定要抓住：数组是经过移位的，那么中间元素的两侧肯定有一边是升序的。
>
> 找到数组A中间位置mid，确定目标数 x 在该数的哪一边。
>
> 1. 如果x == A[mid]时， 找到了返回mid
> 2. 当x大于A[mid]时，分两种情况：
>   - 如果数组开头数A[left]大于A[mid]，说明右半边是升序，因为移位之后，肯定会将大的元素移到左边，此时，如果x > A[right]时，说明x在mid的左半侧
>   - 否则：x位于区间右半侧
> 3. 当x小于mid，分两种情况：
>   - 如果数组开头元素A[left] < A[mid], 说明左边是有序的，此时如果x < A[left], 说明x在区间的右半侧
>   - 否则：x位于区间左半侧

```
class Finder {
public:
    int findElement(vector<int> A, int n, int x)
    {
        // 使用[left, right]区间表示A数组中的所有元素
        int left=0, right = n-1;
        while(left <= right)
        {
            // 找到区间的中间位置，然后用x与中间位置元素比较
            int mid = left + ((right - left) >> 1);

            // 相等代表找到，直接返回中间位置
            if(x == A[mid])
                return mid;

            if(x > A[mid])
            {
                //A[mid] < A[left] 说明右边是有序的，且 x > A[right]说明x在mid左边
                if(A[mid] < A[left] && x > A[right])
                    right=mid-1;
                else
                    left=mid+1;
            }
            else {
                //A[mid]>A[left]说明左边是有序的，且x<A[left]，说明x在mid右边
                if(A[mid] > A[left] && x < A[left])
                    left=mid+1;
                else
                    right=mid-1;
            }
        }

        return -1;
```

```
    }
};
```