柠檬水找零

在柠檬水摊上，每一杯柠檬水的售价为 5 美元。

顾客排队购买你的产品，（按账单 `bills` 支付的顺序）一次购买一杯。

每位顾客只买一杯柠檬水，然后向你付 5 美元、10 美元或 20 美元。你必须给每个顾客正确找零，也就是说净交易是每位顾客向你支付 5 美元。

注意，一开始你手头没有任何零钱。

如果你能给每位顾客正确找零，返回 `true` ，否则返回 `false` 。

示例 1：

输入：[5,5,5,10,20]

输出：true

解释：

前 3 位顾客那里，我们按顺序收取 3 张 5 美元的钞票。

第 4 位顾客那里，我们收取一张 10 美元的钞票，并返还 5 美元。

第 5 位顾客那里，我们找还一张 10 美元的钞票和一张 5 美元的钞票。

由于所有客户都得到了正确的找零，所以我们输出 true。

示例 2：

输入：[5,5,10]

输出：true

示例 3：

输入：[10,10]

输出：false

示例 4：

输入：[5,5,10,10,20]

输出：false

解释：

前 2 位顾客那里，我们按顺序收取 2 张 5 美元的钞票。

对于接下来的 2 位顾客，我们收取一张 10 美元的钞票，然后返还 5 美元。

对于最后一位顾客，我们无法退回 15 美元，因为我们现在只有两张 10 美元的钞票。

由于不是每位顾客都得到了正确的找零，所以答案是 false。

解题思路：
贪心算法：
由于顾客只可能给你三个面值的钞票，而且我们一开始没有任何钞票，因此我们拥有的钞票面值只可能是 5 美元 10 美元和 20 美元三种。基于此，我们可以进行如下的分类讨论。

5 美元，由于柠檬水的价格也为 5 美元，因此我们直接收下即可。

10 美元，我们需要找回 5 美元，如果没有 5 美元面值的钞票，则无法正确找零。

20 美元，我们需要找回 15 美元，此时有两种组合方式，**一种是一张 10 美元和 5 美元**的钞票，**一种是 3 张 5 美元的钞票**，如果两种组合方式都没有，则无法正确找零。当可以正确找零时，两种找零的方式中我们更倾向于第一种，即如果存在 5 美元和 10 美元，我们就按第一种方式找零，否则按第二种方式找零，因为需要使用 5 美元的找零场景会比需要使用 1010 美元的找零场景多，我们需要尽可能保留 5 美元的钞票。

基于此，我们维护两个变量five 和 ten 表示当前手中拥有的 5 美元和 10美元钞票的张数，从前往后遍历数组分类讨论即可。

```
bool lemonadeChange(int* bills, int billsSize){
    int five=0,ten=0;
    for(int i=0;i<billsSize;i++)
    {
        if(bills[i]==5)
        {
            five++;
        }
        else if(bills[i]==10)
        {
            if(five==0)
                return false;
            five--;
            ten++;
        }
        else //20 元情况
        {
            if(five>0&&ten>0)//有 10 元 5 元零钱优先
            {
                five--;
                ten--;
```

```
            }
            else if(five>=3)//有三张 5 元
            {
                five-=3;
            }
            else
                return false;
        }
    }
    return true;
}
```

5 元由于不需找零，直接收下即可，即不存在 return false 情况

10 元由于必须得有 5 元找零，所以 5 元不存在　返回 return false

20元分两种情况，一是5+10组合　二是5+5+5组合　　二者都不存在

返回 return false

其余情况认为均可正确找零 返回 return true