下面需要將陣列 book[1]~book[9] 初始化為 0，因為剛開始桌面上一張牌也沒有。

```
for(i=1;i<=9;i++)
   book[i]=0;
```

接下來，如果桌面上增加了一張牌面為 2 的牌，那就需要將 book[2] 設定為 1，表示牌面為 2 的牌桌上已經有了。當然如果這張牌面為 2 的牌被拿走後，需要及時將 book[2] 重新設定為 0，表示桌面上已經沒有牌面為 2 的牌了。這樣一來，尋找桌上是否有與打出的牌牌面相同的牌，就不需要再迴圈列舉桌面上的每一張牌了，而只需用一個 if 判斷即可。這一點是不是有點像第 1 章第 1 節的桶子排序的方法呢？具體如下。

```
t=q1.data[q1.head];      //小哼先亮出一張牌
if(book[t]==0)           // 表明桌上沒有牌面為 t 的牌
{
   //小哼此輪沒有贏牌
   q1.head++;            //小哼已經打出一張牌，所以要把打出的牌出佇列
   s.top++;
   s.data[s.top]=t;      //再把打出的牌放到桌上，即入堆疊
   book[t]=1;            //標記桌上現在已經有牌面為 t 的牌
}
```

OK，演算法的實作講完了，下面列出完整的程式碼，如下：

```
#include <stdio.h>
struct queue
{
   int data[1000];
   int head;
   int tail;
};

struct stack
{
   int data[10];
   int top;
};

int main()
{
   struct queue q1,q2;
   struct stack s;
   int book[10];
   int i,t;

   //初始化佇列
   q1.head=1; q1.tail=1;
   q2.head=1; q2.tail=1;
   //初始化堆疊
   s.top=0;
   //初始化用來標記的陣列，用來標記哪些牌已經在桌上
   for(i=1;i<=9;i++)
```

```
    book[i]=0;

//依次向佇列插入 6 個數
//小哼手上的 6 張牌
for(i=1;i<=6;i++)
{
    scanf("%d",&q1.data[q1.tail]);
    q1.tail++;
}

//小哈手上的 6 張牌
for(i=1;i<=6;i++)
{
    scanf("%d",&q2.data[q2.tail]);
    q2.tail++;
}

while(q1.head<q1.tail && q2.head<q2.tail ) //當佇列不為空的時候執行迴圈
{
    t=q1.data[q1.head];//小哼出一張牌

    //判斷小哼目前打出的牌是否能贏牌
    if(book[t]==0) //表明桌上沒有牌面為 t 的牌
    {
        //小哼此輪沒有贏牌
        q1.head++; //小哼已經打出一張牌，所以要把打出的牌出佇列
        s.top++;
        s.data[s.top]=t; //再把打出的牌放到桌上，即入堆疊
        book[t]=1; //標記桌上現在已經有牌面為 t 的牌
    }
    else
    {
        //小哼此輪可以贏牌
        q1.head++;//小哼已經打出一張牌，所以要把打出的牌出佇列
        q1.data[q1.tail]=t;//緊接著把打出的牌放到手中牌的末尾
        q1.tail++;
        while(s.data[s.top]!=t) //把桌上可以贏得的牌依次放到手中牌的末尾
        {
            book[s.data[s.top]]=0;//取消標記
            q1.data[q1.tail]=s.data[s.top];//依次放入佇列尾
            q1.tail++;
            s.top--; //堆疊中少了一張牌，所以堆疊頂要減 1
        }
        //收回牌面為 t 的牌
        book[s.data[s.top]]=0;
        q1.data[q1.tail]=s.data[s.top];
        q1.tail++;
        s.top--;
    }

    t=q2.data[q2.head]; //小哈出一張牌
    //判斷小哈目前打出的牌是否能贏牌
    if(book[t]==0) //表明桌上沒有牌面為 t 的牌
    {
        //小哈此輪沒有贏牌
        q2.head++; //小哈已經打出一張牌，所以要把打出的牌出佇列
```

```
            s.top++;
            s.data[s.top]=t; //再把打出的牌放到桌上，即入堆疊
            book[t]=1; //標記桌上現在已經有牌面為 t 的牌
        }
        else
        {
            //小哈此輪可以贏牌
            q2.head++;//小哈已經打出一張牌，所以要把打出的牌出佇列
            q2.data[q2.tail]=t;//緊接著把打出的牌放到手中牌的末尾
            q2.tail++;
            while(s.data[s.top]!=t) //把桌上可以贏得的牌依次放到手中牌的末尾
            {
                book[s.data[s.top]]=0;//取消標記
                q2.data[q2.tail]=s.data[s.top];//依次放入佇列尾
                q2.tail++;
                s.top--;
            }
            //收回牌面為 t 的牌
            book[s.data[s.top]]=0;
            q2.data[q2.tail]=s.data[s.top];
            q2.tail++;
            s.top--;
        }
    }

    if(q2.head==q2.tail)
    {
        printf("小哼 win\n");
        printf("小哼目前手中的牌是");
        for(i=q1.head;i<=q1.tail-1;i++)
            printf(" %d",q1.data[i]);
        if(s.top>0) //如果桌上有牌則依次輸出桌上的牌
        {
            printf("\n 桌上的牌是");
            for(i=1;i<=s.top;i++)
                printf(" %d",s.data[i]);
        }
        else
            printf("\n 桌上已經沒有牌了");
    }
    else
    {
        printf("小哈 win\n");
        printf("小哈目前手中的牌是");
        for(i=q2.head;i<=q2.tail-1;i++)
            printf(" %d",q2.data[i]);
        if(s.top>0) //如果桌上有牌則依次輸出桌上的牌
        {
            printf("\n 桌上的牌是");
            for(i=1;i<=s.top;i++)
                printf(" %d",s.data[i]);
        }
        else
            printf("\n 桌上已經沒有牌了");
    }
```

```
    getchar();getchar();
    return 0;
}
```

可以輸入以下資料進行驗證。

```
2 4 1 2 5 6
3 1 3 5 6 4
```

執行結果是：

```
小哈 win
小哈目前手中的牌是 6 5 2 3 4 1
桌上的牌是 3 4 5 6 2 1
```

接下來你需要自己設計一些測試資料來檢驗你的程式。在設計測試資料的時候你要考慮各種情況，包括各種極端情況。透過設計測試資料來檢測我們的程式是否「健壯」是非常重要的，如果你的程式可以透過任何一組測試資料，才表示你的程式是完全正確的。

如果你設計一些測試資料來驗證的話，會發現我們剛才的程式碼其實還是有問題的。比如遊戲可能無法結束。就是小哼和小哈可以永遠玩下去，誰都無法贏得對方所有的牌。請你自己想一想如何解決遊戲無法結束的問題。