E：朝田诗乃的哥德巴赫猜想

这道题考察了判断一个数是不是素数的方法

素数的定义是，一个大于1的自然数，除了1和它本身外，不能被其他自然数整除。

所以这就可以导出这个题的思路：从2开始到 $\frac{n}{2}$ ，枚举每一个数 a ，对于 a 做如下判断：2 到 $sqrt(a)$ 之间是否有a的因子，如果没有，那么说明 a 是质数，同理可以判断 $n-a$ 是否为质数 。判断结束后，若两者均为素数，输出即可。

```
#include <stdio.h>
int main()
{
    int n, t;
    scanf("%d", &t);
    for (int i = 1; i <= t; i++)
    {
        scanf("%d", &n);
        int ans = 2; //从2开始，依次判断是否满足ans与n-ans是否均为素数
        for (; ans <= n / 2; ans++)
        {
            int flag = 1; //记录ans与n-ans是否均为素数
            for (int j = 2; j * j <= ans; j++) //判断ans是否为素数
            { //若ans不为素数，则ans的最小质因子一定小于等于sqrt(ans)，即j*j<=ans
                if (ans % j == 0) //若可被j整除，则不是素数，标记并跳出循环
                {
                    flag = 0;
                    break;
                }
            }
            for (int j = 2; j * j <= n - ans; j++) //判断n-ans是否为素数
            {
                if ((n - ans) % j == 0) //同理
                {
                    flag = 0;
                    break;
                }
            }
            if (flag == 1) //均为素数则跳出循环并输出
                break;
        }
        printf("%d %d\n", ans, n - ans);
    }
    return 0;
}
```

同时，我们也有一些更好的方法，例如统计所有的素数，并把他们都存下来

下面使用的统计质数的方法叫做埃拉托斯特尼筛法。虽然名字听起来很高级，但其实大家可能都或多或少有了解过这种筛法。即 在一张纸上写出 2 到 n 的数字，然后从 2 开始，依次划去纸上所有是 2 的倍数的数，然后寻找下一个没被划去的数（它便是下一个质数），再次划去纸上所有它的倍数……不断进行，直至到 n，这样纸上剩下的数就都是质数了。

```c
#include <stdio.h>
int main()
{
    int Not_Prime[10020] = {}, Prime[10020] = {}, cnt = 0;
    // Not_Prime表示判定一个数是否不是质数，若NotPrime[i]=1表示i不是质数，0则是
    // Prime为质数表，cnt为质数的计数器
    for (int i = 2; i <= 10000; i++) //统计10000以内的质数
    {
        if (Not_Prime[i]) //如果不是质数，则跳过，继续判断下一个数
            continue;
        Prime[cnt++] = i;
        //如果是质数，那么Prime[cnt] = i，记录这个质数。在这之后，cnt++，计数器加一
        for (int j = 2; i * j <= 10000; j++)
            Not_Prime[i * j] = 1; //给这个质数的倍数标记为非质数
    }
    int t, n;
    scanf("%d", &t);
    for (int i = 1; i <= t; i++)
    {
        scanf("%d", &n);
        int flag = 1; //记录是否找到
        for (int j = 0; j < cnt && flag; j++) //若找到，即flag为0时跳出循环
        { //遍历质数表，
            for (int k = cnt - 1; Prime[j] + Prime[k] >= n && flag; k--)
            //若找到，即flag为0时跳出循环。
            { // cnt为质数的数量，而质数从表Prime[0]开始存储，所以k的初始值为cnt-1
                if (Prime[j] + Prime[k] == n)
                {
                    printf("%d %d\n", Prime[j], Prime[k]);
                    flag = 0;
                }
            }
        }
    }
    return 0;
}
```