A 比较好懂的题

看似很难（实则不难叭

只要按照要求自定义出函数

之后根据题意一步步做就可以了

预备知识：取模运算的性质，用以解决运算过程中可能越界的问题

(a + b) % p = (a % p + b % p) % p;

(a - b) % p = (a % p - b % p) % p;

(a * b) % p = (a % p * b % p) % p;

更多知识可以自行百度搜索“取模运算”

```
#include<stdio.h>
#define MAX 101 // 定义数组最大长度

int main(){
    int num[MAX];
    int n, k, l, r;
    scanf("%d%d", &n, &k);
    int i, temp, N_l_r, M_l_r;
    for(i = 0; i < n; i++)
        scanf("%d", &num[i]);
    while(k--){
        scanf("%d%d", &l, &r);
        temp = 0;
        for(i = l; i <= r ;i++)
            temp = (temp + num[i]) % n;
        N_l_r = temp;
        temp = 1;
        for(i = l; i <= r; i++)
            temp = (temp * num[i]) % n;
        M_l_r = temp;
        if(N_l_r > M_l_r){
            temp = N_l_r;
            N_l_r = M_l_r;
            M_l_r = temp;
        }
        temp = 0; // 0异或一个数后等于它本身，因此将temp初始化为0
        for(i = N_l_r; i <= M_l_r; i++)
            temp = temp ^ num[i];
        printf("%d\n", temp);
    }
    return 0;
}
```

B CWD的质数查找

方法一：

质数查找有许多方法，对应时间复杂度也不同

首先最简单最容易理解的一种方法

直接贴代码

```
#include<stdio.h>
int isPrime(int n){
    // 判断一个数是否为质数
    int i;
    if(n == 2 || n == 3)
        return 1;
    if(n % 2 == 0)
        return 0;
    for(i = 3; i * i <= n; i += 2) // 此处简单的简化时间复杂度，只需i*i<=n即可
        if(n % i == 0)
            return 0;
    return 1;
}
int main() {
    int n;
    scanf("%d", &n);
    while(1){
        n++;
        // 寻找n之后的下一个质数
        if(isPrime(n)){
            printf("%d", n);
            break;
        }
    }
    return 0;
}
```

方法二：

// 不算什么好办法，只是供大家参考了解（嘻嘻嘻

预备知识：一个数不为质数（即合数），那么它一定具有质数的因子

因此我们只需要从质数因子中来筛选，而无需一点点遍历递增筛选

```
#include<stdio.h>
#define MAX 100000  // 定义本题中可能出现的质数的最大个数

int Prime[MAX] = {2};   // 存储所有质数
int ptr = 3;    // 从小到大生成所有质数，作为循环“指针”
                // 定义此时生成质数所在的位置，无需从头再开始遍历
```

```
                // 此简化方法可能更适合多组数据输入的情况
int cnt = 1;    // 表示此时出现的质数个数

int isPrime(int n){
    // 判断一个数是否为质数
    if(n == 2 || n == 3)
        return 1;
    int i;
    for(i = 0; i < cnt; i++)    // 判断当前质数中是否有此数的因子
        if(n % Prime[i] == 0)
            return 0;
    return 1;
}
void createPrime(int n){
    for(; ptr < n; ptr += 2)
        if(isPrime(ptr))
            Prime[cnt++] = ptr;
}

int main(){
    int n;
    scanf("%d", &n);
    while(1){
        n++;
        createPrime(n);
        if(isPrime(n))
            break;
    }
    printf("%d", n);
    return 0;
}
```

C 第k大的数

方法一：

本题由于每个数据均有范围

因而可以记录每个数出现的次数

最后从大到小遍历即可

```
#include<stdio.h>
#define MAX 100001

int cnt[MAX] = {0}; // 数组初始化，cnt用于记录每个元素出现的次数

int main(){
    int n, k, temp;
    scanf("%d", &n);
    while(n--){
```

```
        scanf("%d", &temp);
        cnt[temp]++;
    }
    scanf("%d", &k);
    n = MAX;
    // 寻找第k大的元素
    while(k > 0){
        n--;
        if(cnt[n] > 0)
            k--;
    }
    printf("%d %d", n, cnt[n]);
    return 0;
}
```

方法二：

// 即使“快排”，但涉及到排序，时间复杂度相比上种方法emmm

本题的排序部分为大家介绍快排算法

使用快排的三步骤

- 引入标准库 #include<stdlib.h>
- 自己定义一个函数 int cmp(const void *a, const void *b){return *(int *)a - *(int *)b;}
- 主函数中调用 qsort(数组名， 数组长度， 每个元素数组大小， 第2步中自己定义的函数);

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10000001
int cmp(const void *a, const void *b) {
    //return *(int *)a - *(int *)b;
    if (*(int *)a > *(int *)b) return 1;
    else if (*(int *)a == *(int *)b) return 0;
    else return -1;
}
int num[MAX];   // 数组太大需要定义为全局数组
int main() {
    int i, n, k;
    scanf("%d", &n);
    for(i = 0; i < n; i++)
        scanf("%d", &num[i]);
    qsort(num, n, sizeof(num[0]), cmp); // 对num数组从小到大排序
    scanf("%d", &k);
    int cnt;    // cnt表示此元素出现的次数
    i = n - 1;
    // 寻找第k大的元素
    while(k > 0) {
        cnt = 1;
        for(i -= 1; i >= 0 && num[i] == num[i + 1]; i--)
            cnt++;
        // 记录第k大出现的次数
        k--;
    }
```

```
    printf("%d %d", num[i + 1], cnt);
    return 0;
}
```

写在后面：虽然这种排序算法称为快速排序算法，但是它的效率并不是最高的，甚至在某些情况下也会出现“翻车”的情况，所以大家不要盲目认为这就是最好的一种排序算法。

有能力的同学可以继续深究……

D 小明去春游

本题要求算组合数

本质是一个递归过程

阅读代码后应该都能理解

```
#include<stdio.h>
int fun(int m, int n) {
    if(m < n || m == 0)
        return 0;   // no solution
    if(m == n || n == 0)
        return 1;   // only one solution
    if(n == 1)
        return m;   // select one from total
    return fun(m - 1, n - 1) + fun(m - 1, n);   // selected and not selected
}
int main() {
    int n, m;
    scanf("%d%d", &m, &n);
    printf("%d", fun(m, n));
    return 0;
}
```