```cpp
//日期：2018/ 时间：
/*
考虑如何从一个无序数组中求出第K大的数。最直接的想法是对数组排序，然后直接取出第K个元素。但是这样的复杂度为O(nlogn)
        而随机选择算法的期望复杂度为O(n)
随机选择算法的原理：
    对A[left,right]执行一次randPartition函数后，主元左侧的元素个数就是确定的，且它们都小于主元。
    假设此时主元是A[p],那么A[p]就是A[left,right]中的第i-left+1大的数。
    令M=i-left+1，如果K==M成立，那么第K大的数就是主元A[p];
                    如果K<M成立，那么第K大的数在主元左侧(即A[left,p-1]中第K大的数)，往左递归即可;
                    如果K>M成立，那么第K大的数在主元右侧(即A[p+1,right]中第K-M大的数)，往右递归即可。
    算法以left==right作为递归边界，返回A[left]
代码如下： int randSelect(int A[],int left,int right,int K)
*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <algorithm>
using namespace std;

const int maxn = 100010;
int A[maxn],n;          //A存放所有整数，n为其个数

//快速排序产生复杂度为O(n^2) ，见p144.规避方法是：在A[left,right]中随机选取一个主元，
//因此不妨生成一个范围在[left,right]内的随机数p，然后以A[p]作为主元进行划分。
//具体做法是：将A[p]与A[left]交换，然后按照原先的Partition函数的写法即可
int randPartition(int A[],int left,int right){
    int p = (int)(round(1.0*rand()/RAND_MAX * (right-left) + left));
    //round()函数负责将double型变量四舍五入取整
    swap(A[p],A[left]);         //include <algorithm>

    int temp = A[left];     //将A[left]存放至临时变量temp想象一串数组中有一个空位置
    while(left < right){
        while(left < right && A[right] > temp) right--;
        A[left]=A[right];
        while(left < right && A[left] <= temp ) left++;
        A[right]=A[left];
    }
    A[left] = temp;
    return left;
}

//随机选择算法，从A[left,right]中返回第K大的数
int randSelect(int A[],int left,int right,int K){
    if(left == right)   return A[left];     //边界
    int p = randPartition(A,left,right);    //划分后主元的位置为p
    int M = p-left+1;                       //A[p]是A[left,right]中的第M大
    if(K == M)  return A[p];                //找到第K大的数
    if(K < M){      //第K大的数在主元左侧
        return randSelect(A,left,p-1,K);    //往主元左侧找第K大
```

```
51      } else{
52          return randSelect(A,p+1,right,K-M); //往主元右侧找第K-M大
53      }
54  }
55
56  /*应用:
57  给定一个整数集合，集合中的整数各不相同，现在要将他们分为两个子集合，使得这两个集合的并为全集，交为空集。
58  在两个子集合的元素个数n1与n2之差的绝对值|n1-n2|尽可能小的情况下，要求他们各自的元素之和S1与S2差的绝对值|S1-S2|尽可能大。求|S1-S2|
59
60  思路:
61  n为偶数，两个集合元素个数分别为n/2。n为奇数，两个集合元素个数分别为n/2和n/2+1
62  显然，为使|S1-S2|尽可能大，最直接的思路就是先排序，取前n/2个元素为前一个子集，后面的元素为另一个子集。时间复杂度为O(nlogn)
63
64  而更优的做法是使用随机选择算法，求集合中第n/2大的数字。这样就把数组分成两部分。而不用管数组内部元素的排序方法。
65  */
66  int main(){
67      srand((unsigned)time(NULL));        //初始化随机数种子
68      //sum和sum1记录所有整数之和与切分后前n/2个元素之和
69      int sum = 0,sum1 = 0;
70      scanf("%d",&n);                     //整数个数
71      for(int i=0;i<n;i++){
72          scanf("%d",&A[i]);              //输入整数
73          sum += A[i];                    //累计所有整数之和
74      }
75      randSelect(A,0,n-1,n/2);            //寻找第n/2大的数，并进行切分
76      for(int i=0;i<n/2;i++){
77          sum1 += A[i];                   //累计较小的子集合中元素之和
78      }
79
80      printf("%d\n",(sum - sum1) -sum1);  //求两个子集合的元素和之差
81
82      return 0;
83  }
84
85
```