# 运行出错

右键（是项目而不是项目解决方案）属性-》常规-》平台工具集-》从v140改成120

# 2、// 面试题28：

对称的二叉树

// 题目：请实现一个函数，用来判断一棵二叉树是不是对称的。如果一棵二叉树和

// 它的镜像一样，那么它是对称的。

这道题很简单，我是按照原文中的抄的，按照递归的方法，比较这两个节点是否为空，如果为空，就返回true，如果只有一个节点为空，就返回false，如果值不相等，就返回false，否则就递归访问子节点

# 面试题37、序列化和反序列化二叉树

// 面试题37：序列化二叉树

// 题目：请实现两个函数，分别用来序列化和反序列化二叉树。

序列化二叉树：把一棵二叉树按照某种遍历方式的结果以某种格式保存为字符串，从而使得内存中建立起来的二叉树可以持久保存。

反序列化二叉树：根据某种遍历顺序得到的序列化字符串结果str，重构二叉树

# 最小的k个数

// 面试题40：最小的k个数

// 题目：输入n个整数，找出其中最小的k个数。例如输入4、5、1、6、2、7、3、8

// 这8个数字，则最小的4个数字是1、2、3、4。

使用multiset，里面的greater<int>意思是一个选项，是从大到小排序

我没有思考，只看了本题的答案，首先找前k大的数，可以用快排的思想，如果排序好后，这个数的下表刚好是k，那么就直接输出下标是0~k-1的数就可以了，这种方法的时间复杂度是n，但是会改变数组中的元素，第二种方法是使用最大堆，新建一个大小为k的辅助数组，然后每次从数组中取出一个元素，和这个数组中最大的比较，如果比最大的小，那么就放入数组，将这个数删除，但是如果新建一个最大堆复杂度会很高，所以就用了set中的multiset，multiset的特点是能存储相同的数据，下面我写一下源代码，这个基本上就是答案里面的，我只是再熟悉一下过程：

#incldue<cstdio>

#include<stack>

#include<vector>

#include<set>

//#include<incfunctional>这里我写错了

#include<functional>

using namespace std;

typedef multiset<int,std::greater<int>> intSet;

typedef multiset<int,std::greater<int>>:: iterator multiIter;

void getLeastNumbers const vector<int>& numbers,intSet& leastNum,int k){

leastNumber.clear(); //这里我一开始忘记写了

if(k<1||numbers.size()<k)

return;

vector<int>::const\_int iter=numbers.begin();

for(;iter!=numbers.end();iter++){

if(leastNum.size()<k)

//leastNum.insert(iter); 这里我写错了

leastNum.insert(\*iter);

else{

//multiIter=leastNum.begin();//这里我写错了，这里只声明了还没新建对象

multiIter intSetGreast=leastNum.begin();

if(\*intSetGreast >\*iter){

leastNum.erase(multiIter);//这里要注意，erase只需要传指针不用传值

//leastNum.insert(iter);这里我写错了，应该是\*iter

leastNum.insert(\*iter);

}

}

}

}

# 4、求中位数

// 面试题41：数据流中的中位数

// 题目：如何得到一个数据流中的中位数？如果从数据流中读出奇数个数值，那么

// 中位数就是所有数值排序之后位于中间的数值。如果从数据流中读出偶数个数值，

// 那么中位数就是所有数值排序之后中间两个数的平均值。

vector.size() 求vector的大小

vector.push\_back(<T>) 将一个元素插入到vector尾部

vector.pop\_back(<T>) 从vector尾部删除元素

push\_heap(vector.begin(),vector.end(),less<T>()) 将vector末尾元素加入大顶堆中

pop\_heap(vector.begin(),vector.end(),less<T>()) 将大顶堆最大值放到末尾，然后将vector.begin()到vector.end()-1重新排成大顶堆

如果是小顶堆，less改成greater