我选择的是C3插入排序以及Hoare公理化方法的反向证明法。 首先是问题描述，这里我不仅描述了问题还简单描述了插入排序的算法流程：给定一个未排序数组分为两部分，每次从未排序数组取出第一个元素，从后往前扫描已排序数组，插入已排序数组的第一个小于等于它的元素后的位置。

下面根据问题描述建立程序规范，包括输入和输出断言：输入限制了数组长度，输出明确了数组的升序规则：

这里沿用了课件的符号u perm，也解释了两个符号的含义。

下面将算法用JAVA代码和符合Hoare公理学的伪代码描述，其中它需要转换成while型程序。根据上述的伪代码绘制了流程图程序和结构化程序。

化简结构化程序，合并汇点，根据程序流程图可以添加如下的8个断点和6条通路，为每条通路分别建立引理，后续需要证明各个引理的正确性。

下面为先前的伪代码添加断言：为了避免后续的逻辑推导符合过多，我定义了四个逻辑函数：

这里进一步说明了perm的含义，original表示循环前的数组，表示两个数组的排列相同，它的作用是防止每一轮插入操作后，当前数组和插入前的数组元素不一致（包括值和数量），sorted表示数组在一个范围内已经排序，equal表示j位置之后的两个值相等，同时排除了一个特殊情况，change表示取当前A的副本对某一个位置进行元素设置。

同时我也对内层、外层循环均添加了不变式。

外层的含义可以理解为：循环后元素不变，且1..i-1范围内有序

内层的含义可以理解为：1-j有序 j+1-i有序，j+1到i范围的数都大于key，key是原始数组i位置的数，如果把当前数组j+1位置的数替换为key，那么它和原来的数组的排列相同

循环结束的条件就是1-n均排好序且元素排列和原始数组相同，最后的z仅用来保存结果

下面开始证明各个部分：

可以发现右式代入定义后是恒成立的

证明外层循环的正确性可以进行条件分解，将满足循环做一种情况，不满足循环做另一种情况。这里先证明满足循环的情况。

要证明这种情况，需要证明下面的几个情况：

1. 进入内层循环前的正确性，这里我们先把左右都展开，将j=i-1代入，会发现右式的sorted可以由左式推导，且equal恒成立，同时有右式的另一个个sorted恒成立，代入k的值得到对于所有的这个条件成立，因为推导过程未修改数组，所以下面的change条件成立，因而(2)成立。
2. 下面证明内层循环的正确性，这里将两个内层循环正确的条件添加至输入断言中：

先将左、右式展开，这里的set和change不同，就是直接修改A数组了。因为1-j有序，且set没有改变1-j的值，所以1:j-1有序。因为执行set后A[j+1]=A[j]，因此sorted[j:j+1]成立，又因为左式sorted[j+1:i]成立，所以sorted[j:i]成立。

同时题目中给出了A[j]>key，同时我们将A[j+1]设为了A[j],利用左式的条件得到了右式对于所有的一部分成立，因为内层循环没有对key和i进行赋值，所以key=original[i]也成立。

下面证明元素的排序没有发生变化，这里分两种情况：

1. 此时因为set导致equal成立，PPT稿
2. 另一种情况下，当左式成立时有A[j+2]=A[j+1]

执行set后A[j]和A[j+1]的值相等，之前是change j+1，现在是changej，相当于是原始的两个元素交换位置

下面证明两个出口的返回外循环的正确性：

1. 展开左、右式，此时根据change和set的定义可以证明执行set也就是赋值操作后没有破坏元素的排列

同时由左式的equal和sorted得到的条件可知执行set后A[1:i]依然有序

1. 依然先将左、右式展开

从左式获取到排序条件，将k=j+1代入得到如下的条件，和上一条证明类似，此时的change和赋值语句的含义相同

最后是外层的退出和一个恒成立的赋值语句的证明，

通过上述证明，我们证明了插入排序算法的部分正确性。