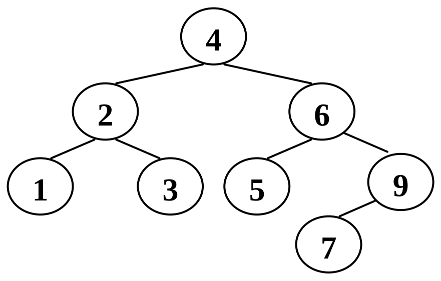
p.136 4.16

Show the result of inserting 2, 1, 4, 5, 9, 3, 6, 7 into an initially empty AVL tree.

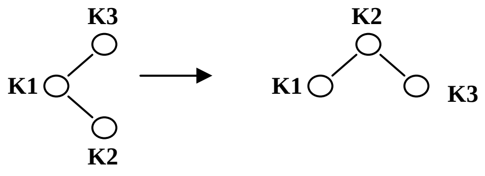


p.136 4.22

Write the functions to perform the double rotation without the inefficiency of doing two single rotations.

```
#ifndef   _AvlTree_H
#define   _AvlTree_H
struct   AvlNode;
typedef   struct   AvlNode   *Position;
typedef   struct   AvlNode   *AvlTree;
/* function declarations are omitted */
#endif   /* _AvlTree_H */
struct   AvlNode   {
    ElementType   Element;
    AvlTree   Left, Right;
    int   Height;
}

static Position   DoubleRotateWithLeft( Position   K3 )
{   /* Do the left—right double rotation.   K3 is the trouble finder. */
     Position   K1, K2;
     K1=K3->Left;        /* mark parent */
     K2=K1->Right;       /* mark trouble maker */
     K1->Right=K2->Left;
     K3->Left=K2->Right;
     K2->Left=K1;
     K2->Right=K3;       /* finish setting links */
     K1->Height=Max( Height(K1->Left), Height(K1->Right) ) + 1;
     K3->Height=Max( Height(K3->Left), Height(K3->Right) ) + 1;
     K2->Height=Max( K1->Height, K3->Height ) + 1; /* finish setting heights */
     return   K2;    /* K2 is the new root */
}
```

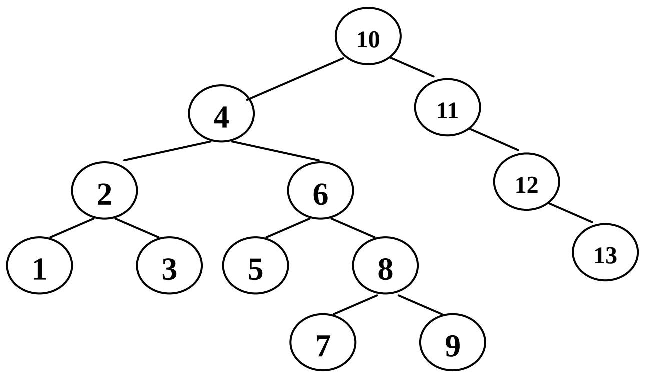


```
static Position   DoubleRotateWithRight( Position   K1 )
{    /* Do the right--left double rotation.   K1 is the trouble finder. */
     Position   K2, K3; /* Similar to the above function */
     K3=K1->Right;
     K2=K3->Left;
     K1->Right=K2->Left;
     K3->Left=K2->Right;
     K2->Left=K1;
     K2->Right=K3;
     K1->Height=Max( Height(K1->Left), Height(K1->Right) ) + 1;
     K3->Height=Max( Height(K3->Left), Height(K3->Right) ) + 1;
     K2->Height=Max( K1->Height, K3->Height ) + 1;
     return   K2;
}
```

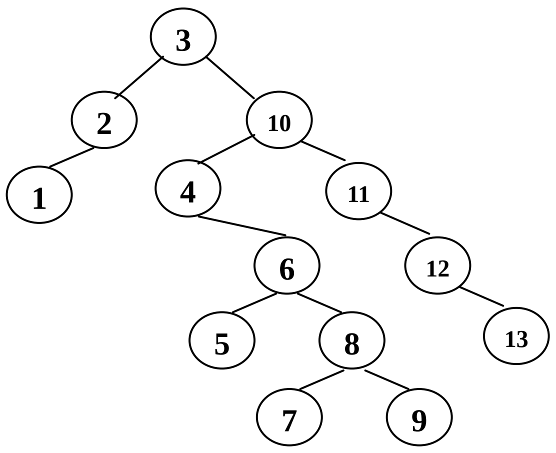
p.136   4.23

Show the result of accessing the keys 3, 9, 1, 5 in order in the splay tree in Figure 4.61.

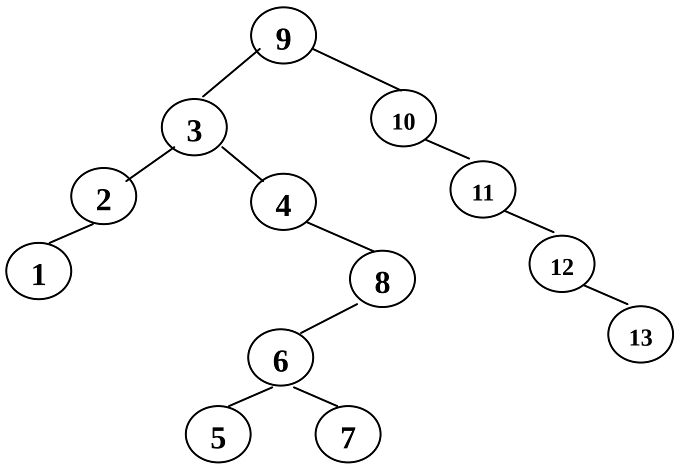
Figure 4.61



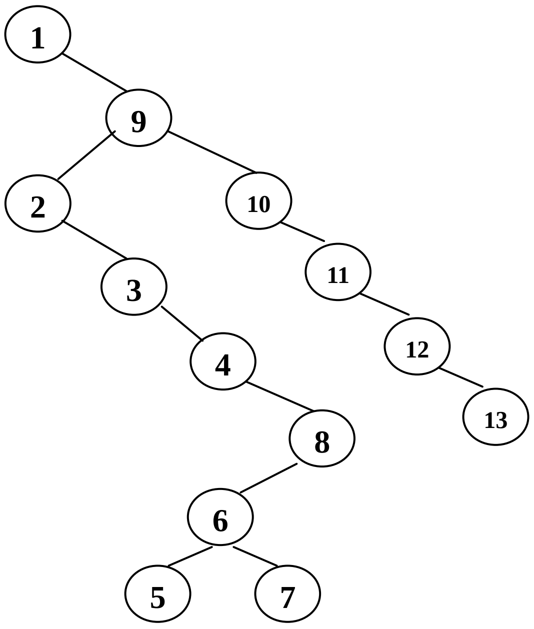
Result for 3:



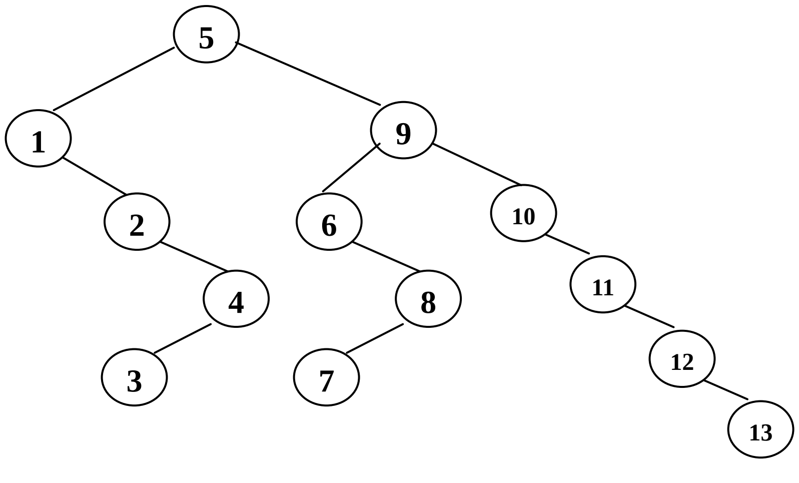
Result for 9:



Result for 1:



Result for 5:


5
1
9
2
6
10
4
8
11
3
7
12
13