BST AVL Struct

```
template <class T>
class TreeNode {
private:
       T _item;
        Tre eNo de< T>* _left;
        Tre eNo de< T>* _right;
        //Could include
        Tre eNo de< T>* _parent;
       int _height;
public:
        Tre eNode(T x) {
          _left = _right = NULL;
          _item = x;
          _height = 0; };
          friend Binary Sea -
rch Tre e<T >;
};
```

_item = value

_height = height of the node (will have to calculate)

_left , _right = default set to NULL in public constructor.

Search Max,Min

```
template <class T>
T Binary Sea rch Tre e<T >:: -
sea rch Max() {
          Tre eNo de< T>*
current = _root;
          while (curre nt- >_r -
ight) {
                       current =
curren t-> _right;
           }
            return curren t-> -
_item;
}
```

Search Max,Min (cont)

```
> template <class T>
T BinarySearchTree<T>::searchMin() {
    TreeNode<T>* current = _root;
    while (current->_left) {
    current = current->_left;
    }
    return current->_item;
}
```

Successor

```
template <class T>
T Binary Sea rch Tre e<T >:: -
suc ces sor(T x){
          Tre eNo de< T>* root =
_root;
         T succ = NULL;
          while (root != NULL) {
             if (x < root-> -
_item){
                       succ =
root-> _item;
                       root =
root-> _left;
               }
                else{
                       root =
root-> _right;
               }
       }
        return succ;
}
```

exist() // iterative

```
//Iterative
template <class T>
bool Binary Sea rch Tre e<T -
>:: exist(T x) {
                Tre eNo de< T>*
nd = _root;
                while (nd !=
NULL) {
                                  i
f (nd->_item == x) return true;
                                  i
f (nd->_item > x) nd = nd-
>_left;
                                  e
lse nd = nd->_r ight;
              }
               return false;
}
```

exit() // Recursive

```
template <class T>
bool Binary Sea rch Tre e<T -
>:: exi st( Tre eNo de< T>
Node,T x) {
                 // RECURSIVE
                  exists ( _root
, value x )
                if (_root ==
null) return false;
                if (_root -
>_item == x) return true;
                if (_root -
>_item > x)
                          return
exists ( _root- >left, x);
                else exists (
_root- >right, x );
}
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

LR RL Rotation

```
//LR Rotation
template <class T>
TreeNo de< T>Binary Sea rch -
Tre e<T >:: _le ftr igh tRo tat -
ion (Tr eeN ode <T> node)
{
node-> _left = _right Rot ati -
on( nod e-> _left);
node = _leftR ota tio n(n ode);
return node;
}
//RL Rotation
template <class T>
TreeNo de< T>Binary Sea rch -
Tre e<T >:: _ri ght lef tRo tat -
ion (Tr eeN ode <T> node) {
node-> _right = _leftR ota tio -
n(n ode ->_ right);
node = _right Rot ati on( node);
return node;
}
```

PreOrderPrint

```
template <class T>
void Binary Sea rch Tre e<T -
>:: _pr eOr der Pri nt( Tre eNo -
de< T>* node) {
           if (!node) return;
           cout << node-> _item
<< " ";
            _pr eOr der Pri -
nt( nod e-> _left);
            _pr eOr der Pri -
nt( nod e-> _ri ght);
}
```

InOrderPrint

```
template <class T>
void Binary Sea rch Tre e<T> ::
_inOrd erP rin t(T ree Nod e<T -
>* node) {
           if (!node) return;
            _in Ord erP rin -
t(n ode ->_ left);
           cout << node-> _item
<< " ";
            _in Ord erP rin -
t(n ode ->_ right);
}
```

PostOrderPrint

```
template <class T>
void Binary Sea rch Tre e<T >::
_postO rde rPr int (Tr eeN ode -
<T>* node) {
         if (!node) return;
          _po stO rde rPr int -
(no de- >_l eft);
          _po stO rde rPr int -
(no de- >_r ight);
         cout << node-> _item <<
" ";
}
```

Height

```
template <class T>
int Binary Sea rch Tre e<T >:: -
hei ght (Tr eeN ode <T>* node) {
return node == NULL ? -1 :
(node- >_h eight);
}
template <class T>
int Binary Sea rch Tre e<T >:: -
_ca lhe igh t(T ree Nod e<T >*
node) {
     int leftHeight = -1;
```

Height (cont)

```
>    int rightHeight = -1;
   if (node != NULL) {
    if (node->_left != NULL) {
      leftHeight = _calheight(node->_left);
    }
    if (node->_right != NULL) {
     rightHeight = _calheight(node->_right);
    }
   }
   return max(leftHeight, rightHeight) + 1;
}
//Height from TA
template <class T>
void TreeNode<T>::rectifyHeight()
{
 int left = _left ? _left->_height : -1;
 int right = _right ? _right->_height : -1;
//Height Value
 _height = (left > right ? left : right) + 1;
}
```

Insert

```
//Task 1 and 6
template <class T>
TreeNo de< T>Binary Sea rch -
Tre e<T >:: _in ser t(T ree -
Nod e<T> current, T x) {
       if (curre nt- >_item > x)
{
            if (curre nt- >_left)
                 cur ren t-> -
_left = _inser t(c urr ent ->_ -
left, x);
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Insert (cont)

```
>       else {
        current->_left = new TreeNode<T>(x);
        _size++;
       }
    }
    else if (x > current->_item) {
     if (current->_right)
       current->_right = _insert(current->_-
right, x);
      else{
       current->_right = new TreeNode<T>
(x);
       _size++;
      }
    }
    else
//item already exists, dont need do anything
     return current;
```

Code From TA// Rotation

```
//-1 if no children, st bf will
work
int left = curren t-> _left ?
curren t-> _le ft- >_h eight : -
1;
int right = curren t-> _right ?
curren t-> _ri ght ->_ height :
-1;
curren t-> _height = (left >
right ? left : right) + 1;

    //GET DIFF IN height
   if (abs(left - right) > 1){

if (left > right){
```

Code From TA// Rotation (cont)

```
>       int LLH = current->_left->_left ?
current->_left->_left->_height : 0;

      int LRH = current->_left->_right ?
current->_left->_right->_height : 0;
   if (LLH < LRH){
   current->_left = _leftRotation(current->_-
left);
   }
   current = _rightRotation(current);
}
else {
   int RLH = current->_right->_left ? current-
>_right->_left->_height : 0;
   int RRH = current->_right->_right ?
current->_right->_right->_height : 0;
   if (RRH < RLH){
   current->_right = _rightRotation(current-
>_right);
   }
   current = _leftRotation(current);
 }
}
//go through tree and rect height
  current->rectifyHeight();
  return current;
}
```

LL RR Rotation

```
//LL Rotation
template <class T>
TreeNo de< T>Binary Sea rch -
Tre e<T >:: _le ftR ota tio -
n(T ree Nod e<T> node)
{
TreeNo de< T>* nd;
nd = node-> _left;
node-> _left = nd->_r ight;
nd->_right = node;
nd->_h eight = calhei ght (nd);
node-> _height = calhei ght -
(node);
return nd;

/*
*
*Codes from tutori als horizon
*copyr ights to -https: //a -
lgo rit hms.tu tor ial hor izo -
n.c om/ avl -tr ee- ins ertion/
*
//*/
//Tree Nod e<T >* nd = node-> -
_right;
//Tree Nod e<T >* T2 = nd-
>_left;
//nd-> _left = node;
//node ->_ right = T2;
//nd-> _height = calhei ght -
(nd);
//node ->_ height = calhei ght -
(node);
//return nd;
}
//RR Rotation
```


Sponsored by ApolloPad.com
Everyone has a novel in them. Finish Yours!
https://apollopad.com

LL RR Rotation (cont)

```
> template <class T>
TreeNode<T> BinarySearchTree<T>::_rightRotation(TreeNode<T> node)
{
TreeNode<T>* nd;
nd = node->_right;
node->_right = nd->_left;
nd->_left = node;
nd->_height = calheight(nd);
node->_height = calheight(node);
return nd;
/*
*
*Codes from tutorialshorizon
*copyrights to - https://algorithms.tutorialhorizon.com/avl-tree-insertion/
*
*
//
TreeNode<T>* nd = node->_left;
TreeNode<T>* T2 = nd->_right;
nd->_right = node;
node->_left = T2;
nd->_height = calheight(nd);
node->_height = calheight(node);
return nd;*/
}
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com