Consider following implementation of BST for this lab:

```
template <class T>
class BST{
  //inner class available only for class BST
  class TNode{
    public:
      T data;
      TNode *left, *right;
      TNode(T d){
        data=d;
        left=right=NULL;
      }
  };
  TNode *root;
public:
  BST(){ root=NULL;    }
  void add(T d){  root=add(root, d);
  }
  TNode* add(TNode *t, T d){
    if (t==NULL)
      t=new TNode(d);
    else if (t->data==d)     return t;
    else if (t->data>d)
      t->left=add(t->left, d);
    else
      t->right=add(t->right, d);
    return t;
  }
  void inOrder(){ inOrder(root);
  cout<<'\n';     }
  void inOrder(TNode *t){
    if (t){
      inOrder(t->left);
      cout<<t->data<<' ';
      inOrder(t->right);
    }
  }
};
int main(){
  srand((unsigned int)time(0));
  BST <int> tree;
  int i, val;
  int array[15];
  for (i=0;i<10;i++){
    do{
      val=rand()%100;
    }while (isExist(array, val, i));
    array[i]=val;
    tree.add(val);
  }
  tree.inOrder();
  //tree.preOrder();
  //cout<<"Height:"<<tree.height()<<'\n';
  //cout<<"Sum:"<<tree.sum()<<'\n';
  /*cout<<"Event Element Exist:";
  if (tree.isEvenElementExist())
    cout<<"Yes\n";
  else
    cout<<"No\n";*/
  return 0;
}
```

Task 1: Provide recursive implementation of function to find height. A file "tree.txt" contains 15 values. Create BST object for integers. Read values from file add into object of BST by calling "***add***" function. Call height function for your BST.

Task 2: Provide recursive implementation of sum function. Sum function has to add values of all nodes ?

Task 3: Provide recursive implementation of "***isEvenElementExist***" to check whether or not any even element exist in BST?

******************** Enjoy Reeeeeeeeeeeeeeeeeeeeeeeecursion ********************