Exercises - Binary Search Trees

1. Implement a BST that doesn't have a separate Node class, as the example I gave in the class did. You may follow this template:

```
class BTree
{
public:
  int key;  //data in the node
  BTree *left;   // Pointer to the left subtree.
  BTree *right;  // Pointer to the right subtree.
  BTree();
  ~BTree();
  void insert(int key);   //insert a new node at a leaf position with the given int data
  BTree *search(int key); //return NULL if no node has given int value, or a pointer to the node that ha
  void destroy();         //clean up the whole tree
  void preOrderPrint();
  void inOrderPrint();
  void postOrderPrint();
};
//To test your code
int main(){
  BTree* bt=new BTree;
  bt->insert(12);
  bt->insert(4);
  bt->insert(10);
  bt->insert(21);
  bt->insert(13);
  bt->insert(3);
  bt->insert(15);
  bt->insert(22);
  bt->preOrderPrint();
  bt->inOrderPrint();
  bt->postOrderPrint();
  cout<<endl;
  bt->destroy();
}
```

2.For a BST, implement the folowing functions:

```
BTree* findMax(); // returns pointer to node with maximum key value
BTree* findMin(); // returns pointer to node with minimum key value
int countNodes(); // count the total number of nodes in the tree
int countLeafNodes(); // count the number of leaf nodes in the tree
int findDepth(); //find the depth of the tree
```