Group 17 Worksheet 30
Tanner England, Eddie Fox, Trevor Worthey, Lok Chan, Catherine Smith

```
struct BSTIterator {
    struct DynArr   *stk;
    struct BSTree   *tree;
};
void BSTIteratorInit (struct BSTree *tree, struct BSTIterator *itr) {
        itr->stk = dyNew(20); //create dyn array capacity 20
        assert(itr->stk); //make sure it was allocated properly (probably redundant)
        itr->tree = tree;
}

int BSTIteratorHasNext (struct BSTIterator * itr) {
        Node *n;
        if(dyStackIsEmpty(itr->stk)) // if stack is empty perform slideLeft on root
                _slideLeft(itr->tree->root);
        else {
                n = dyStackTop(itr->stack); //let n be top of stack
                dyStackPop(itr->stack); //pop topmost element.
                _slideLeft(n->right); // slideLeft on right child of n
        }
        if(!dyStackIsEmpty(itr->stack)) // return true if stack is not empty
                return 1;
        else
                return 0; //iterator does not have next since stack is empty
}



TYPE BSTIteratorNext (struct BSTIterator *itr) {
        return dyStackTop(itr->stack)->value; //return value of top of stack
}



void _slideLeft(struct Node *cur, struct BSTIterator *itr)
{
        While(cur != null)
          {
                  dynArrayPush(Itr->stk, cur->val);
                  cur = cur->left;
           }
}
```