struct BSTIterator {

struct DynArr \*stk;

struct BSTree \*tree; };

void BSTIteratorInit (struct BSTree \*tree, struct BSTIterator \*itr) {

itr->stk = createDynArr(10);  
 assert(itr->stk != 0)

itr->tree = tree;

}

int BSTIteratorHasNext (struct BSTIterator \* itr) {

Node \*n;

if(isEmptyDynArr(itr->stk)) {

\_slideLeft(itr->tree->root); }

else{

n = topDynArr(itr->stk);

popDynArra(itr->stk)

\_slideLeft(n->right);

}

if(!isEmptyDynArr(itr->stk))

return 1;

else return 0;

}

TYPE BSIteratorNext(struct BSIterator \*itr) {

return(topDynArr(itr->stk))->val;

}

void \_slideLeft(struct Node \*cur, struct BSTIterator \*itr) {

While(cur != null) {

dynArrayPush(Itr->stk, cur->val); cur = cur->left; } }