**Binary search tree**

**1.**

int isInTheTree(BinNode\* node, int key) {

int rank = 0;

while (node != NULL && node->key != key )

{

if (node->key <= key)

node = node->right;

else

node = node->left;

rank++;

}

if (node == NULL)

return -1;

if (node->key == key)

return rank;

}

**O(n) זמן ריצה:**

**Log n במקרה סטנדרטי:**

**2.**

int isInTheTreeRec(BinNode\* node, int key) {

if (key == NULL || node == NULL)

return -1;

int sum=0;

if (node->key == key)

return sum;

if (node->key <= key )

sum = isInTheTreeRec(node->right, key);

if (node->key > key)

sum = isInTheTreeRec(node->left, key);

if (sum == -1)

return -1;

return sum+1;

}

**O(n) זמן ריצה:**

**Log n במקרה סטנדרטי:**

**3.**

int const N = 10;

struct Stack

{

BinNode\* arr[N];

int size;

};

void stackInit(struct Stack& S)

{

S.size = 0;

}

int empty(struct Stack& S)

{

return S.size == 0;

}

void push(struct Stack& S, BinNode\* node)

{

if (S.size == N)

{

printf("Stack Full! \n");

}

else

{

S.arr[S.size] = node;

S.size++;

}

}

BinNode\* pop(struct Stack& S)

{

if (empty(S))

{

printf("stack is empty");

return NULL;

}

S.size--;

return S.arr[S.size];

}

int sumoftree(BinNode\* node) {

Stack s;

stackInit(s);

if (node == NULL)

return 0;

BinNode\* p;

int sum = 0;

push(s, node);

while (!empty(s)) {

p = pop(s);

sum += p->key;

if (p->right)

push(s,p->right);

if (p->left)

push(s, p->left);

}

return sum;

}

int sumOfLittle(BinNode\* node, int key) {

int sum = 0;

BinNode\* root = node;

while (root != NULL) {

while (root != NULL && root->key <= key) {

if (root->key <= key)

sum += root->key;

sum += sumoftree(root->left);

root = root->right;

}

while (root != NULL && root->key > key)

root = root->left;

}

return sum;

}

void insert(BinNode\*& root, int key)

{

if (root == NULL) {

root = newNode(key);

return;

}

if (root->key >= key) //put in left

if (root->left == NULL)

root->left = newNode(key);

else insert(root->left, key);

else //put in right

if (root->right == NULL)

root->right = newNode(key);

else insert(root->right, key);

}

**Main to check**

void main() {

BinNode\* root = newNode(8);

insert(root, 3);

insert(root, 4);

insert(root, 10);

insert(root, 9);

insert(root, 11);

insert(root, 35);

insert(root, 10);

printf("%d\n", isInTheTreeRec(root,9));

printf("%d\n", isInTheTree(root,40));

printf("%d", sumOfLittle(root,5));

}