**Exercise 5**

**Name:** Dolev Mishali.

**ID:** 206603060.

**Github:** <https://github.com/DolevMishali/hw-5.git>

**BST.C**

#define ALLOC(p,n) (p = malloc(n\*sizeof \*p);)

#include "BST.h"

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

#include <stdbool.h>

void initBST(BST\* mgBST)

{

mgBST->root = NULL;

}

void insertBST(BST\* bst, int value)

{

if(bst->root == NULL)

{

bst->root = calloc(1, sizeof(TreeNode));

assert(bst->root);

bst->root->element = value;

return;

}

if (value <= bst->root->element)

if (bst->root->left == NULL)

{

bst->root->left = calloc(1, sizeof(TreeNode));

assert(bst->root->left);

bst->root->left->element = value;

}

else

insertBST(&bst->root->left, value);

if (value > bst->root->element)

{

if (bst->root->right == NULL)

{

bst->root->right = calloc(1, sizeof(TreeNode));

assert(bst->root->right);

bst->root->right->element = value;

}

else

insertBST(&bst->root->right, value);

}

}

void printTreeInorder(BST\* bst)

{

if(bst->root != NULL)

{

printTreeInorder(&bst->root->left);

printf("%d,",bst->root->element);

printTreeInorder(&bst->root->right);

}

}

void destroyBST(BST\* bst)

{

while (1)

{

if (bst->root == NULL)

{

break;

}

else if (bst->root != NULL && bst->root->left != NULL)

{

destroyBST(&bst->root->left);

}

else if (bst->root != NULL && bst->root->right != NULL)

{

destroyBST(&bst->root->right);

}

else

{

free(bst->root);

bst->root = NULL;

return;

}

}

}

int findIndexNFromLast(BST\* bst, int N)

{

static int temp = 0;

static int count = 0;

if (bst->root != NULL)

{

findIndexNFromLast(&bst->root->right,N);

count++;

if (count == N)

{

temp = bst->root->element;

return temp;

}

findIndexNFromLast(&bst->root->left, N);

}

return temp;

}

**BST.h**

#ifndef \_BST\_ADT\_H

#define \_BST\_ADT\_H

typedef struct TreeNode TreeNode;

struct TreeNode

{

TreeNode\* left;

TreeNode\* right;

int element;

};

typedef struct BST

{

TreeNode\* root;

}BST;

void initBST(BST\* bst);

void insertBST(BST\* bst, int value);

void printTreeInorder(BST\* bst);

void destroyBST(BST\* bst);

int findIndexNFromLast(BST\* bst, int N);

#endif // !\_BST\_ADT\_H

**Main.c**

#define ALLOC(p,n) (p = malloc(n\*sizeof \*p);)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

#include "BST.h"

void main()

{

BST\* mgBST = (BST\*)malloc(sizeof(BST));

assert(mgBST);

int highNum = 0;

initBST(mgBST);

insertBST(mgBST, 5);

insertBST(mgBST, 2);

insertBST(mgBST, 1);

insertBST(mgBST, 1);

insertBST(mgBST, 2);

insertBST(mgBST, 4);

insertBST(mgBST, 5);

insertBST(mgBST, 8);

insertBST(mgBST, 8);

insertBST(mgBST, 8);

insertBST(mgBST, 9);

insertBST(mgBST, 9);

//printTreeInorder(mgBST);

//destroyBST(mgBST);

//highNum=findIndexNFromLast(mgBST,6);

}

**ScreenShots:**

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

**תמונה שמכילה טקסט, מסך, צילום מסך

התיאור נוצר באופן אוטומטי**