CS513, Data Structures LAB

Assignment 3

Maximum Marks: 100 Time: 3 Hours

(5 Marks: Indentation, 5 Marks: Commenting Code, 5 Marks: Meaningful variable names)

August 20, 2022

struct bstNode{

int key;

int size;

struct bstNode *leftChild;

struct bstNode *leftChild;};

typedef struct bstNode bstNode;

typedef struct bstNode *bstNodePtr;

void createBST(bstNodePtr *root){ *root = NULL;}

bstNodePtr getTreeNode() *//Allocate a node dynamically*

void displayBST(bstNodePtr root, char *fileName) *// Use graphviz to display tree graphically*

5 points—bstNodePtr bstSearch(bstNodePtr root, int data) - O(h)*// returns the pointer of the node having key value equal to data (successful search) or returns NULL*

10 points—int bstInsert(bstNodePtr *root, int data) - O(h)

40 points—int bstDelete(bstNodePtr *root, int data) - O(h)

30 points—bstNodePtr kthElement(bstNodePtr *root, int k) - O(h)*// returns the pointer of the node having the k-th largest key value or returns NULL*

struct bstNode{

int key;

int size;

struct bstNode *leftChild;

struct bstNode *leftChild;};

typedef struct bstNode bstNode;

typedef struct bstNode *bstNodePtr;

void createBST(bstNodePtr *root){ *root = NULL;}

bstNodePtr getTreeNode() *//Allocate a node dynamically*

void displayBST(bstNodePtr root, char *fileName) *// Use graphviz to display tree graphically*

5 points—bstNodePtr bstSearch(bstNodePtr root, int data) - O(h)*// returns the pointer of the node having key value equal to data (successful search) or returns NULL*

10 points—int bstInsert(bstNodePtr *root, int data) - O(h)

40 points—int bstDelete(bstNodePtr *root, int data) - O(h)

30 points—bstNodePtr kthElement(bstNodePtr *root, int k) - O(h)*// returns the pointer of the node having the k-th largest key value or returns NULL*

struct bstNode{

int key;

int size;

struct bstNode *leftChild;

struct bstNode *leftChild;};

typedef struct bstNode bstNode;

typedef struct bstNode *bstNodePtr;

void createBST(bstNodePtr *root){ *root = NULL;}

bstNodePtr getTreeNode() *//Allocate a node dynamically*

void displayBST(bstNodePtr root, char *fileName) *// Use graphviz to display tree graphically*

5 points—bstNodePtr bstSearch(bstNodePtr root, int data) - O(h)*// returns the pointer of the node having key value equal to data (successful search) or returns NULL*

10 points—int bstInsert(bstNodePtr *root, int data) - O(h)

40 points—int bstDelete(bstNodePtr *root, int data) - O(h)

30 points—bstNodePtr kthElement(bstNodePtr *root, int k) - O(h)*// returns the pointer of the node having the k-th largest key value or returns NULL*

struct bstNode{

int key;

int size;

struct bstNode *leftChild;

struct bstNode *leftChild;};

typedef struct bstNode bstNode;

typedef struct bstNode *bstNodePtr;

void createBST(bstNodePtr *root){ *root = NULL;}

bstNodePtr getTreeNode() *//Allocate a node dynamically*

void displayBST(bstNodePtr root, char *fileName) *// Use graphviz to display tree graphically*

5 points—bstNodePtr bstSearch(bstNodePtr root, int data) - O(h)*// returns the pointer of the node having key value equal to data (successful search) or returns NULL*

10 points—int bstInsert(bstNodePtr *root, int data) - O(h)

40 points—int bstDelete(bstNodePtr *root, int data) - O(h)

30 points—bstNodePtr kthElement(bstNodePtr *root, int k) - O(h)*// returns the pointer of the node having the k-th largest key value or returns NULL*