**LABORATORY ASSIGNMENT 4**

**NAME: AKSHAT SRIVASTAV**

**REGISTRATION ID: 19BCE0811**

**Data Structure and algorithms :**

**Q). Write down a C/C++/C#/JAVA/Python/MATLAB/R Program to Implement Binary Search Tree Using Linked List. Your program must include the following functions:**

**i) INSERT (Data Item), ii) REMOVE (Data Item), and iii) SEARCH (Data Item).**

**C CODE:**

***include*** *<stdio.h>*

***#include*** *<stdlib.h>*

***typedef******struct*** *node {*

***int*** *data;*

***struct*** *node \*left;*

***struct*** *node \*right;*

***struct*** *node \*parent;*

*}node;*

***typedef******struct*** *binary\_search\_tree {*

*node \*root;*

*}binary\_search\_tree;*

*node\* new\_node(****int*** *data) {*

*node \*n = malloc(****sizeof****(node));*

*n->data = data;*

*n->left = NULL;*

*n->right = NULL;*

*n->parent = NULL;*

***return*** *n;*

*}*

*binary\_search\_tree\* new\_binary\_search\_tree() {*

*binary\_search\_tree \*t = malloc(****sizeof****(binary\_search\_tree));*

*t->root = NULL;*

***return*** *t;*

*}*

*node\* minimum(binary\_search\_tree \*t, node \*x) {*

***while****(x->left != NULL)*

*x = x->left;*

***return*** *x;*

*}*

***void*** *insert(binary\_search\_tree \*t, node \*n) {*

*node \*y = NULL;*

*node \*temp = t->root;*

***while****(temp != NULL) {*

*y = temp;*

***if****(n->data < temp->data)*

*temp = temp->left;*

***else***

*temp = temp->right;*

*}*

*n->parent = y;*

***if****(y == NULL) //newly added node is root*

*t->root = n;*

***else******if****(n->data < y->data)*

*y->left = n;*

***else***

*y->right = n;*

*}*

***void*** *transplant(binary\_search\_tree \*t, node \*u, node \*v) {*

***if****(u->parent == NULL) //u is root*

*t->root = v;*

***else******if****(u == u->parent->left) //u is left child*

*u->parent->left = v;*

***else*** *//u is right child*

*u->parent->right = v;*

***if****(v != NULL)*

*v->parent = u->parent;*

*}*

***void*** *delete(binary\_search\_tree \*t, node \*z) {*

***if****(z->left == NULL) {*

*transplant(t, z, z->right);*

*free(z);*

*}*

***else******if****(z->right == NULL) {*

*transplant(t, z, z->left);*

*free(z);*

*}*

***else*** *{*

*node \*y = minimum(t, z->right); //minimum element in right subtree*

***if****(y->parent != z) {*

*transplant(t, y, y->right);*

*y->right = z->right;*

*y->right->parent = y;*

*}*

*transplant(t, z, y);*

*y->left = z->left;*

*y->left->parent = y;*

*free(z);*

*}*

*}*

***void*** *inorder(binary\_search\_tree \*t, node \*n) {*

***if****(n != NULL) {*

*inorder(t, n->left);*

*printf("%d\n", n->data);*

*inorder(t, n->right);*

*}*

*}*

***int*** *main() {*

*binary\_search\_tree \*t = new\_binary\_search\_tree();*

*node \*a, \*b, \*c, \*d, \*e, \*f, \*g, \*h, \*i, \*j, \*k, \*l, \*m;*

*a = new\_node(10);*

*b = new\_node(20);*

*c = new\_node(30);*

*d = new\_node(100);*

*e = new\_node(90);*

*f = new\_node(40);*

*g = new\_node(50);*

*h = new\_node(60);*

*i = new\_node(70);*

*j = new\_node(80);*

*k = new\_node(150);*

*l = new\_node(110);*

*m = new\_node(120);*

*insert(t, a);*

*insert(t, b);*

*insert(t, c);*

*insert(t, d);*

*insert(t, e);*

*insert(t, f);*

*insert(t, g);*

*insert(t, h);*

*insert(t, i);*

*insert(t, j);*

*insert(t, k);*

*insert(t, l);*

*insert(t, m);*

*delete(t, a);*

*delete(t, m);*

*inorder(t, t->root);*

***return*** *0;*

*}*