Ex. No. 6 **Binary Search Tree**

Date:

**Aim**

To insert and delete nodes in a binary search tree.

**Algorithm**

1. Create a structure with key and 2 pointer variable left and right.

2. Read the node to be inserted.

If (root==NULL)

root=node

else if (root->keykey)

root->right=NULL

else

Root->left=node

3. For Deletion

if it is a leaf node

Remove immediately

Remove pointer between del node & child

if it is having one child

Remove link between del node&child

Link delnode is child with delnodes parent

If it is a node with a children

Find min value in right subtree

Copy min value to delnode place

Delete the duplicate

4. Stop

**Program**

/\* Binary Search Tree \*/

#include <stdio.h>

#include <stdlib.h>

struct node

{

int key;

struct node \*left;

struct node \*right;

};

struct node \*newNode(int item)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

printf("%d ", root->key);

inorder(root->right);

}

}

struct node\* insert(struct node\* node, int key)

{

if (node == NULL)

return newNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else

node->right = insert(node->right, key);

return node;

}

struct node \* minValueNode(struct node\* node)

{

struct node\* current = node;

while (current->left != NULL)

current = current->left;

return current;

}

struct node\* deleteNode(struct node\* root, int key)

{

struct node \*temp;

if (root == NULL)

return root;

if (key < root->key)

root->left = deleteNode(root->left, key);

else if (key > root->key)

root->right = deleteNode(root->right, key);

else

{

if (root->left == NULL)

{

temp = root->right;

free(root);

return temp;

}

else if (root->right == NULL)

{

temp = root->left;

free(root);

return temp;

}

temp = minValueNode(root->right);

root->key = temp->key;

root->right = deleteNode(root->right, temp->key);

}

return root;

}

void main()

{

struct node \*root = NULL;

root = insert(root, 50);

root = insert(root, 30);

root = insert(root, 20);

root = insert(root, 40);

root = insert(root, 70);

root = insert(root, 60);

root = insert(root, 80);

printf("Inorder traversal of the given tree \n");

inorder(root);

printf("\nDelete 20\n");

root = deleteNode(root, 20);

printf("Inorder traversal of the modified tree \n");

inorder(root);

printf("\nDelete 30\n");

root = deleteNode(root, 30);

printf("Inorder traversal of the modified tree \n");

inorder(root);

printf("\nDelete 50\n");

root = deleteNode(root, 50);

printf("Inorder traversal of the modified tree \n");

inorder(root);

}

**Output:**

**Result**

Thus nodes were inserted and deleted from a binary search tree.