#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*right\_child;

struct node \*left\_child;

};

struct node\* new\_node(int x){

struct node \*temp;

temp = malloc(sizeof(struct node));

temp->data = x;

temp->left\_child = NULL;

temp->right\_child = NULL;

return temp;

}

struct node\* search(struct node \* root, int x){

if (root == NULL || root->data == x)

return root;

else if (x > root->data)

return search(root->right\_child, x);

else

return search(root->left\_child, x);

}

struct node\* insert(struct node \* root, int x){

if (root == NULL)

return new\_node(x);

else if (x > root->data)

root->right\_child = insert(root->right\_child, x);

else

root -> left\_child = insert(root->left\_child, x);

return root;

}

struct node\* find\_minimum(struct node \* root) {

if (root == NULL)

return NULL;

else if (root->left\_child != NULL)

return find\_minimum(root->left\_child);

return root;

}

struct node\* delete(struct node \* root, int x) {

if (root == NULL)

return NULL;

if (x > root->data)

root->right\_child = delete(root->right\_child, x);

else if (x < root->data)

root->left\_child = delete(root->left\_child, x);

else {

if (root->left\_child == NULL && root->right\_child == NULL){

free(root);

return NULL;

}

else if (root->left\_child == NULL || root->right\_child == NULL){

struct node \*temp;

if (root->left\_child == NULL)

temp = root->right\_child;

else

temp = root->left\_child;

free(root);

return temp;

}

else {

struct node \*temp = find\_minimum(root->right\_child);

root->data = temp->data;

root->right\_child = delete(root->right\_child, temp->data);

}

}

return root;

}

void inorder(struct node \*root){

if (root != NULL)

{

inorder(root->left\_child);

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

inorder(root->right\_child);

}

}

int main() {

struct node \*root;

int x;

root = new\_node(20);

insert(root, 5);

insert(root, 1);

insert(root, 15);

insert(root, 9);

insert(root, 7);

insert(root, 12);

insert(root, 30);

insert(root, 25);

insert(root, 40);

insert(root, 45);

insert(root, 42);

printf("\n elements after creating Binary search tree\n");

inorder(root);

printf("\n");

printf("\n element to search");

scanf("%d",&x);

if(search(root,x)==NULL)

printf("\n %d element is not found",x);

else

printf("\n %d element is found",x);

root = delete(root, 1);

root = delete(root, 40);

root = delete(root, 45);

root = delete(root, 9);

printf("\n elements after deletion\n");

inorder(root);

printf("\n");

return 0;

}

O/P:

elements after creating Binary search tree

1 5 7 9 12 15 20 25 30 40 42 45

element to search4

4 element is not found

elements after deletion

5 7 12 15 20 25 30 42