```
//C Program To Demonstrate Deletion & Height Calculation Of A BST

#include <stdio.h>
#include <stdlib.h>

struct node
{
	int key;
	struct node *left, *right;
};

// function to create a new BST node
struct node *newNode(int item)
{
	struct node *temp =  (struct node *)malloc(sizeof(struct node));
	temp->key = item;
	temp->left = temp->right = NULL;
	return temp;
}

// function to do inorder traversal of BST
void inorder(struct node *root)
{
	if (root != NULL)
	{
	inorder(root->left);
	printf("%d ", root->key);
	inorder(root->right);
	}
}

/* function to insert a new node with given key in BST */
struct node* insert(struct node* node, int key)
{
	/* If the tree is empty, return a new node */
	if (node == NULL) return newNode(key);

	/* Otherwise, recur down the tree */
	if (key < node->key)
	node->left  = insert(node->left, key);
	else
	node->right = insert(node->right, key);
```

```
	/* return the (unchanged) node pointer */
	return node;
}

struct node * minValueNode(struct node* node)
{
	struct node* current = node;

	/* loop down to find the leftmost leaf */
	while (current->left != NULL)
	current = current->left;

	return current;
}


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

	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)
	{
	struct node *temp = root->right;
	free(root);
	return temp;
	}
	else if (root->right == NULL)
	{
	struct node *temp = root->left;
	free(root);
	return temp;
	}
```

```
        struct node* temp = minValueNode(root->right);

        root->key = temp->key;

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

int maxDepth(struct node* node)
{
  if (node==NULL)
        return 0;
  else
  {
        /* compute the depth of each subtree */
        int lDepth = maxDepth(node->left);
        int rDepth = maxDepth(node->right);

        /* consider the greater one */
        if (lDepth > rDepth)
        return(lDepth+1);
        else return(rDepth+1);
  }
}

int main(){
  int ch;
  struct node *root = NULL;
  int a=0;
  while(a==0){
        printf("\nPress 1 to add\nPress 2 to delete a node\nPress 3 to display the
in-order\nPress 4 to display height\nPress 5 to EXIT\nEnter Choice : ");
        scanf("%d",&ch);
        switch(ch){
        case 1: {
        int val;
        printf("\nEnter Value: ");
        scanf("%d",&val);
        root=insert(root,val);
        break;
        }
        case 2: {
```

```
        int val;
        printf("\nEnter Value: ");
        scanf("%d",&val);
        root=deleteNode(root,val);
        break;
        }
        case 3: printf("\n");inorder(root);printf("\n");break;
        case 4: printf("Height of tree is %d\n", maxDepth(root));break;
        case 5: a=1; break;
        default: printf("\nINVALID.");

        }
  }
}
```