//Program to Create & Perform Operations On Binary Search Tree – Visakh Bobby S3R2 34

#include<stdio.h>

#include<stdlib.h>

struct node // node components in binaryST

{

  int info;

  struct node \* lt;

  struct node \* rt;

};

struct node \*root = NULL; //root of the binary tree , initially NULL

//Creating a new node

struct node \* createNode(int info)

{

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

  new->info = info;

  new->lt = NULL;

  new->rt = NULL;

  return new;

}

struct node\* BST\_Insert(struct node\* node , int uinfo)

{ struct node \*new = createNode(uinfo);

  if(node==NULL) //if root is NULL

    return new;

  if (uinfo < node->info)

node->lt = BST\_Insert(node->lt, uinfo); //assigning the left of node

else if (uinfo> node->info)

node->rt = BST\_Insert(node->rt, uinfo); //assigning the rightof node

return node;

}

void inOrder(struct node \*node)

{

  if(root==NULL)

  {

    printf("Tree Is Empty\n");

    return;

  }

  else

  {

    if(node->lt!=NULL)

      inOrder(node->lt);

    printf("%d\t",node->info);

    if(node->rt!=NULL)

      inOrder(node->rt);

  }

}

struct node\* BST\_Minimum(struct node\*root)

{

  if(root->lt!=NULL)

    return BST\_Minimum(root->lt);

  else

    return root;

}

struct node\* BST\_Search(struct node\* root, int key)

{

  if(root== NULL || root->info == key)

    return root;

  else

  {

    if(key<root->info)

      return BST\_Search(root->lt,key);

    else

      return BST\_Search(root->rt,key);

  }

}

struct node\* BST\_Delete(struct node \*node , int delval) //returns delval of type struct node\*

{

  if(node == NULL)

    return NULL;

  else

  { //traversing through to find the postion of the node to be deleted by using BST properties

    if(delval<node->info)

      node->lt = BST\_Delete(node->lt,delval);

    else if(delval>node->info)

      node->rt = BST\_Delete(node->rt,delval);

    //now if val = node->info - element is found , execute else

    else

    {

      if(node->lt == NULL && node->rt == NULL) //no child nodes

      { free(node);

        return NULL;

      }

      else if(node->lt == NULL) //one child at right , so we move the right child up

        node = node->rt;

      else if(node->rt == NULL) //only one child at left , so we move the left child up.

        node = node->lt;

      else //node to be deleted has two child nodes

      {

        //we find minimum node from the right subtree of node to be deleted , exchange that data with current node and call the recursive fn BST\_Delete

        struct node\* temp = BST\_Minimum(node->rt);

        node->info = temp ->info;

        node->rt = BST\_Delete(node->rt,temp->info);

      }

    }

  return node;

  }

}

void main()

{

  struct node\* delNode = NULL;

  int delVal=0;

  int uinfo=0;

  int ch=0;

  int skey=0;

  //struct node\* searchElmt;

  printf("Binary Search Tree Operations :\n 1.Insert To Binary Search Tree\n 2.Delete From Binary Search Tree\n 3.Search In Binary Search Tree\n 4.Display Binary Search Tree(InOrder)\n 5.Exit Menu\n" );

  printf("Enter Your Choice:\n");

  scanf("%d",&ch);

  while(ch>=1 && ch <=4)

  {

    switch(ch)

    {

      case 1 : printf("Enter Element To Be Inserted:\n");

           scanf("%d",&uinfo);

           if(root==NULL)

            root = BST\_Insert(root,uinfo);

           else

            BST\_Insert(root,uinfo);

           break;

      case 2 : printf("Enter Value To Be Deleted:\n");

           scanf("%d",&delVal);

           if(BST\_Delete(root,delVal))

            printf("%d has been Deleted From Tree\n",delVal);

           else

            printf("%d is Not Present In Tree\n");

           break;

      case 3 : printf("Enter the Element To Be Searched:\n");

           scanf("%d",&skey);

           if(BST\_Search(root,skey))

            printf("Element %d Is Found !\n",skey);

           else

            printf("Element %d Is NOT Found!\n",skey);

           break;

      case 4: printf("Binary Search Tree:\n");

          inOrder(root);

          printf("\n");

          break;

      case 5 : exit(0);

    }

    printf("Enter Choice Again:\n");

    scanf("%d",&ch);

  }

}

**Output**

