NAME- Aniket Kalbhor

PRN- 12210601

ROLL- CSB-48

CODE- 5**.Write a menudriven  program in C to perform following  operations  on BST**

**a.Create b. Insert c.Inorderr traversal d.Mirror tree e.Height of the tree f.Level wise display g. Leaf node display h.Exit**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct node {

int data;

struct node \*l;

struct node \*r;

};

struct node \*root = NULL;

struct node\*createBST(int data)

{

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

if (newNode == NULL)

printf("Memory Allocation failed\n");

newNode->data = data;

newNode->l = NULL;

newNode->r = NULL;

return newNode;

}

struct node \*insertNode(struct node\* root, int newdata)

{

if(root == NULL){

root = createBST(newdata);

}

else if (newdata>root->data)

root->r = insertNode(root->r, newdata);

else

root->l = insertNode(root->l, newdata);

return root;

}

void RinOrder(struct node \*root)

{

if(root != NULL){

RinOrder(root->l);

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

RinOrder(root->r);

}

}

void mirrorImage(struct node \*root)

{

if(root != NULL){

struct node \*temp;

mirrorImage(root->l);

mirrorImage(root->r);

temp = root->l;

root->l = root->r;

root->r = temp;

}

}

void levelDisplay(struct node \*root)

{

struct node \*queue[100];

int front = 0, rear = 0;

if(root == NULL)

return;

queue[rear++] = root;

printf("\nLevel order traversal is: ");

while(front < rear){

struct node \*temp = queue[front++];

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

if (front == rear)

printf("\n");

if(temp->l != NULL)

queue[rear++] = temp->l;

if(temp->r != NULL)

queue[rear++] = temp->r;

}

}

int height(struct node \*root)

{

if(root == NULL)

return 0;

else{

int lheight = height(root->l);

int rheight = height(root->r);

if(lheight > rheight)

return lheight+1;

else

return rheight+1;

}

}

void leafNodes(struct node \*root)

{

if(root != NULL){

if(root->l == NULL && root->r == NULL)

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

leafNodes(root->l);

leafNodes(root->r);

}

}

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

{

if (node == NULL)

return NULL;

while(node->l != NULL)

node = node->l;

return node;

}

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

{

struct node \*dtemp;

if(root==NULL)

return root;

else if (key<root->data)

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

else if (key>root->data)

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

else{

if(root->l == NULL){

dtemp = root->r;

free(root);

return dtemp;

}

else if(root->r == NULL){

dtemp = root->l;

free(root);

return dtemp;

}

else{

dtemp = minValueNode(root->r);

root->data = dtemp->data;

root->r = deleteNode(root->r, dtemp->data);

}

}

return root;

}

int main()

{

int choice;

int createdata;

int newdata;

int deletedata;

while(1){

printf("\n1. Create BST\n2. Insert\n3. Delete\n4. InOrder\n5. Mirror Image\n6. Level displaay\n7. Height\n8. Leaf Nodes\n9. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch(choice){

case 1:

printf("Enter the data: ");

scanf("%d", &createdata);

root = createBST(createdata);

break;

case 2:

printf("Enter the data: ");

scanf("%d", &newdata);

root = insertNode(root, newdata);

break;

case 3:

printf("Enter the data: ");

scanf("%d", &deletedata);

root = deleteNode(root, deletedata);

break;

case 4:

RinOrder(root);

break;

case 5:

RinOrder(root);

mirrorImage(root);

printf("\nMirror Image is: ");

RinOrder(root);

break;

case 6:

levelDisplay(root);

break;

case 7:

printf("\nHeight of the tree is: %d\n", height(root));

break;

case 8:

printf("\nLeaf nodes are: ");

leafNodes(root);

break;

case 9:

exit(0);

break;

}

}

}

OUTPUT-

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 1

Enter the data: 25

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 2

Enter the data: 20

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 2

Enter the data: 40

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 2

Enter the data: 10

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 4

10 20 25 40

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 6

Level order traversal is: 25

20 40 10

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 7

Height of the tree is: 3

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 8

Leaf nodes are: 10 40

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 5

10 20 25 40

Mirror Image is: 40 25 20 10

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit

Enter your choice: 8

Leaf nodes are: 40 10

1. Create BST

2. Insert

3. Delete

4. InOrder

5. Mirror Image

6. Level displaay

7. Height

8. Leaf Nodes

9. Exit