









**Program Code :-**

#include <iostream>

using namespace std;

struct node

{

int data;

node \*L;

node \*R;

};

node \*root, \*temp;

int count, key;

class bst

{

public:

void create();

void insert(node \*, node \*);

void disin(node \*);

void dispre(node \*);

void dispost(node \*);

void search(node \*, int);

int height(node \*);

void mirror(node \*);

void min(node \*);

bst()

{

root = NULL;

count = 0;

}

};

void bst::create()

{

char ans;

do

{

temp = new node;

cout << "Enter the data : ";

cin >> temp->data;

temp->L = NULL;

temp->R = NULL;

if (root == NULL)

{

root = temp;

}

else

insert(root, temp);

cout << "Do you want to insert more value :(y/n) : " << endl;

cin >> ans;

count++;

cout << endl;

} while (ans == 'y');

cout << "The Total no.of nodes are : " << count;

}

void bst::insert(node \*root, node \*temp)

{

if (temp->data > root->data)

{

if (root->R == NULL)

{

root->R = temp;

}

else

insert(root->R, temp);

}

else

{

if (root->L == NULL)

{

root->L = temp;

}

else

insert(root->L, temp);

}

}

void bst::disin(node \*root)

{

if (root != NULL)

{

disin(root->L);

cout << root->data << "\t";

disin(root->R);

count++;

}

}

void bst::dispre(node \*root)

{

if (root != NULL)

{

cout << root->data << "\t";

dispre(root->L);

dispre(root->R);

}

}

void bst::dispost(node \*root)

{

if (root != NULL)

{

dispost(root->L);

dispost(root->R);

cout << root->data << "\t";

}

}

void bst::search(node \*root, int key)

{

int flag = 0;

cout << "\nEnter your key : " << endl;

cin >> key;

temp = root;

while (temp != NULL)

{

if (key == temp->data)

{

cout << "KEY FOUND\n";

flag = 1;

break;

}

node \*parent = temp;

if (key > parent->data)

{

temp = temp->R;

}

else

{

temp = temp->L;

}

}

if (flag == 0)

{

cout << "KEY NOT FOUND " << endl;

}

}

int bst::height(node \*root)

{

int hl, hr;

if (root == NULL)

{

return 0;

}

else if (root->L == NULL && root->R == NULL)

{

return 0;

}

cout << endl;

hr = height(root->R);

hl = height(root->L);

if (hr > hl)

{

return (1 + hr);

}

else

{

return (1 + hl);

}

}

void bst::min(node \*root)

{

temp = root;

cout << endl;

while (temp->L != NULL)

{

temp = temp->L;

}

cout << root->data;

}

void bst::mirror(node \*root)

{

temp = root;

if (root != NULL)

{

mirror(root->L);

mirror(root->R);

temp = root->L;

root->L = root->R;

root->R = temp;

}

}

int main()

{

bst t;

int ch;

char ans;

do

{

cout << "\n1) Insert new node\n2)number of nodes in longest path\n3) minimum\n4) mirror\n5) search\n6) inorder\n7) preorder\n8) postorder" << endl;

cin >> ch;

switch (ch)

{

case 1:

t.create();

break;

case 2:

cout << "\n Number of nodes in longest path: " << (1 + (t.height(root)));

break;

case 3:

cout << "\nThe min element is: ";

t.min(root);

break;

case 4:

t.mirror(root);

cout << "\nThe mirror of tree is: ";

t.disin(root);

break;

case 5:

t.search(root, key);

break;

case 6:

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INORDER\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

t.disin(root);

break;

case 7:

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PREORDER\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

t.dispre(root);

break;

case 8:

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POSTORDER\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

t.dispost(root);

break;

}

cout << "\nDo you want to continue (y/n): ";

cin >> ans;

} while (ans == 'y');

return 0;

}

**Program Output: -**

Text

Description automatically generated

Text

Description automatically generated with low confidence

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated