**EX:11 IMPLEMENTATION OF AVL TREEE**

#include <stdio.h>

#include <stdlib.h>

// Define a node

struct node

{

int data;

struct node \* left,\*right;

int height;

};

// Calculate height

int getHeight(struct node \*temp)

{

// Check if temp is NULL, if yes return 0 else return the height

if(temp == NULL)

{

return 0;

}

else

{

return temp->height;

}

}

int max(int a, int b)

{

// Check if the value of a is greater than b. If yes, return a else return b

if(a > b)

{

return a;

}

else

{

return b;

}

}

int getBalanceFactor(struct node \*temp)

{

if(temp == NULL)

{

return 0;

}

else

{

return getHeight(temp->left) - getHeight(temp->right);

}

}

// Create a node

struct node \* newnode(int value)

{

struct node\*newnode;

newnode=malloc(sizeof(struct node));

newnode -> data=value;

newnode -> left = NULL;

newnode -> right = NULL;

newnode ->height =0;

return (newnode);

}

// Right rotate

struct node \*rightRotate(struct node \*y)

{

struct node \*x = y->left;

struct node \*t = x->right;

x->right = y;

y->left = t;

y->height = max(getHeight(y->left), getHeight(y->right)) + 1;

x->height = max(getHeight(x->left), getHeight(x->right)) + 1;

return x;

}

// Left rotate

struct node \*leftRotate(struct node \*x)

{

struct node \*y = x->right;

struct node \*t = y->left;

y->left = x;

x->right = t;

x->height = max(getHeight(x->left), getHeight(x->right)) + 1;

y->height = max(getHeight(y->left), getHeight(y->right)) + 1;

return y;

}

struct node \*insertNode(struct node \*temp, int value)

{

if(temp == NULL)

return newnode(value);

if(value < temp->data)

temp->left = insertNode(temp->left,value);

else if(value > temp->data)

temp->right = insertNode(temp->right,value);

else

return temp;

temp->height = 1 + max(getHeight(temp->left), getHeight(temp->right));

int balance= getBalanceFactor(temp);

if((balance > 1) && (value < temp->left->data))

{

return rightRotate(temp);

}

if((balance < -1) && (value > temp->right->data))

{

return leftRotate(temp);

}

if((balance > 1) && (value > temp->left->data))

{

temp->left = leftRotate(temp->left);

return rightRotate(temp);

}

if((balance < -1) && (value > temp->right->data))

{

temp->right = rightRotate(temp->right);

return leftRotate(temp);

}

return temp;

}

void inorder(struct node \*root)

{

if(root!=NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

int main()

{

struct node \*root = NULL;

int n,ele,i;

printf("\nEnter the number of nodes: ");

scanf("%d",&n);

printf("\nEnter the element: ");

for(i=0;i<n;i++)

{

scanf("%d",&ele);

root = insertNode(root,ele);

}

inorder(root);

return 0;

}

OUTPUT: