**PROGRAM FOR FINDING AVL TREE:**

#include<iostream>

#include<stdlib.h>

#include<string.h>

#include<stdio.h>

using namespace std;

class node

{

public:

char keyword[20];

char meaning[20];

node \*left;

node \*right;

int ht;

node()

{

strcpy(meaning,"");

strcpy(keyword,"");

}

};

class tree

{

private:

node \*root;

tree \*t;

public:

tree()

{

root=NULL;

}

void insert1(char key[20],char mean[20])

{

//root=new node();

root=insert(root,key,mean);

}

node \*insert(node \*T,char word[20],char mean[20]);

void preorder(node \*);

void inorder(node \*);

void postorder(node \*);

void display();

void modify(node \*);

void delet();

node \* In\_Succ(node \*);

void ascending()

{

inorder(root);

}

void descending(node \*);

void search1(node \*);

void search()

{

search1(root);

}

void max\_compare(node \*);

node \*Rotate\_Left(node \*T);

node \*Rotate\_Right(node \*T);

int BF(node \*T);

int height(node \*T);

};

int tree::BF(node\*T)

{

int lh,rh;

if(T==NULL)

return 0;

else

{

if(T->left==NULL)

{

lh=0;

}

else

{

lh=1+T->left->ht;

}

if(T->right==NULL)

{

rh=0;

}

else

{

rh=1+T->right->ht;

}

}

return(lh-rh);

}

node \*tree::Rotate\_Right(node\*x)

{

node\*y;

y=x->left;

x->left=y->right;

y->right=x;

x->ht=height(x);

y->ht=height(y);

return(y);

}

node \*tree::Rotate\_Left(node\*T)

{

node\*temp;

temp=T->right;

T->right=temp->left;

temp->left=T;

T->ht=height(T);

temp->ht=height(temp);

return(temp);

}

int tree::height(node\*T)

{

int left\_height,right\_height;

if(T==NULL)

{

return(0);

}

if(T->left==NULL)

{

left\_height=0;

}

else

{

left\_height=1+(T->left->ht);

}

if(T->right==NULL)

{

right\_height=0;

}

else

{

right\_height=1+(T->right->ht);

}

if(left\_height>right\_height)

return(left\_height);

else

return(right\_height);

}

node \*tree::insert(node \*T,char word[20], char mean[20])

{

if(T==NULL)

{

T=new node();

}

if(strcmp(T->keyword,"")==0)

{

strcpy(T->keyword,word);

strcpy(T->meaning,mean);

T->left=NULL;

T->right=NULL;

}

else

{

if(strcmp(word,T->keyword)>0)

{

T->right=insert(T->right,word,mean);

if(BF(T)==-2)

{

if(strcmp(word,T->right->keyword)>0)

{

T=Rotate\_Left(T);

}

else

{

T->right=Rotate\_Right(T->right);

T=Rotate\_Left(T);

}

}

}

else

{

if(strcmp(word,T->keyword)<0)

{

T->left=insert(T->left,word,mean);

if(BF(T)==2)

{

if((strcmp(word,T->left->keyword)<0))

{

T=Rotate\_Right(T);

}

else

{

T->left=Rotate\_Left(T->left);

T=Rotate\_Right(T);

}

}

}

}

}

T->ht=height(T);

return(T);

}

void tree::modify(node \*head)

{

char data[20],mean[20],flag;

node \*temp=head;

cout<<"\n Enter the word for which u want to modify the meaning";

cin>>data;

cout<<"\n Enter the new meaning";

cin>>mean;

while(temp!=NULL)

{

if(strcmp(temp->keyword,data)==0)

{

strcpy(temp->meaning,mean);

flag=1;

break;

}

if(strcmp(temp->keyword,data)<0)

temp=temp->right;

else

temp=temp->left;

}

if(flag==1)

{

cout<<"\n The keyword is\_\_\_\_" <<temp->keyword<<"\_\_\_ and its modified meaning is-"<<temp->meaning<<"\n";

}

else

{

cout<<"\n The word" <<data<<"is not present in the BST \n";

}

display();

}

node \*tree::In\_Succ(node \*root)

{

node \*trav;

trav=root;

trav=trav->right;

while(trav->left!=NULL)

{

trav=trav->left;

}

return trav;

}

void tree::preorder(node \*root)

{

if(root!=NULL)

{

cout<<endl<<(root)->keyword<<" ! "<<(root)->meaning;

preorder((root)->left);

preorder((root)->right);

}

}

void tree::inorder(node \*root)

{

if(root!=NULL)

{

inorder((root)->left);

cout<<endl<<(root)->keyword<<" ! "<<(root)->meaning;

inorder((root)->right);

}

}

void tree::descending(node \*root)

{

if(root!=NULL)

{

descending((root)->right);

cout<<endl<<(root)->keyword<<" ! "<<(root)->meaning;

descending((root)->left);

}

}

void tree::postorder(node \*root)

{

if(root!=NULL)

{

postorder((root)->left);

postorder((root)->right);

cout<<endl<<(root)->keyword<<" ! "<<(root)->meaning;

}

}

void tree::display()

{

int ch;

do{

cout<<endl<<"Enter ur choice"<<endl;

cout<<"1.Pre-order"<<endl;

cout<<"2.In-order"<<endl;

cout<<"3.Post-order"<<endl;

cout<<"4.Modify "<<endl;

cout<<"5.Search"<<endl;

cout<<"6.Ascending order \n";

cout<<"7.Descending order \n";

cout<<"8.Max Comparisons\n";

cout<<"0.Exit"<<endl;

cin>>ch;

switch(ch)

{

case 1:cout<<"\n KEYWORD"<<" ! "<<"MEANING";

cout<<"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

preorder(root);

break;

case 2:cout<<"\n KEYWORD"<<" ! "<<"MEANING";

cout<<"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

inorder(root);

break;

case 3:cout<<"\n KEYWORD"<<" ! "<<"MEANING";

cout<<"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

postorder(root);

break;

case 4:modify(root);

break;

case 5:search();

break;

case 6:ascending();

break;

case 7:descending(root);

break;

case 8:max\_compare(root);

break;

case 0:exit(0);

break;

default:cout<<"\n Wrong choice";

break;

}

} while(ch!=0);

}

void tree::search1(node \*head) //the function belongs to class 'tree'

{

node \*temp=head; //'head' is pointer to root node

char data[20];

int flag;

cout<<"\n Enter the word u want to search";

cin>>data;

while(temp!=NULL)

{

if(strcmp(temp->keyword,data)==0)

{

flag=1;

break;

}

if(strcmp(temp->keyword,data)<0)

temp=temp->right;

else

temp=temp->left;

}

if(flag==1)

{

cout<<"\n The word---" <<data<<"---is present in the BST \n";

}

else

{

cout<<"\n The word" <<data<<"isnot present in the BST \n";

}

display();

}

void tree::max\_compare(node \*head) //the function belongs to class 'tree'

{

node \*temp=head; //'head' is pointer to root node

char data[20];

int flag,count=0;

cout<<"\n Enter the word u want to search";

cin>>data;

while(temp!=NULL)

{

if(strcmp(temp->keyword,data)==0)

{

count++;

flag=1;

break;

}

if(strcmp(temp->keyword,data)<0)

temp=temp->right;

else

temp=temp->left;

}

if(flag==1)

{

cout<<"\n The max comparisons for the word: "<<count<<" \n";

}

else

{

cout<<"\n The word " <<data<<" is not present in the BST \n";

}

display();

}

int main()

{

tree b;

char key[20],mean[20];

FILE \*fp;

fp=fopen("input.dat","r");

if(fp==NULL)

{

cout<<"\n File opening error.";

exit(0);

}

cout<<"\n Reading keywords and meaning from the dictionary";

while(!feof(fp))

{

fscanf(fp,"%s%s",key,mean);

b.insert1(key,mean);

fflush(fp);

}

b.display();

fclose(fp);

return 0;

}

**OUTPUT**:

Reading keywords and meaning from the dictionary

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

1

KEYWORD ! MEANING

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

v ! vv

e ! ee

c ! cc

b ! bb

d ! dd

m ! mm

f ! ff

r ! rr

y ! yy

x ! xx

z ! zz

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

2

KEYWORD ! MEANING

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b ! bb

c ! cc

d ! dd

e ! ee

f ! ff

m ! mm

r ! rr

v ! vv

x ! xx

y ! yy

z ! zz

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

3

KEYWORD ! MEANING

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b ! bb

d ! dd

c ! cc

f ! ff

r ! rr

m ! mm

e ! ee

x ! xx

z ! zz

y ! yy

v ! vv

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

4

Enter the word for which u want to modify the meaning

m

Enter the new meaning

mother

The keyword is\_\_\_\_m\_\_\_ and its modified meaning is-mother

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

5

Enter the word u want to search

r

The word---r---is present in the BST

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

6

b ! bb

c ! cc

d ! dd

e ! ee

f ! ff

m ! mother

r ! rr

v ! vv

x ! xx

y ! yy

z ! zz

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

7

z ! zz

y ! yy

x ! xx

v ! vv

r ! rr

m ! mother

f ! ff

e ! ee

d ! dd

c ! cc

b ! bb

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

8

The max comparisons for the word: 1

Enter ur choice

1.Pre-order

2.In-order

3.Post-order

4.Modify

5.Search

6.Ascending order

7.Descending order

8.Max Comparisons

0.Exit

0