/\*

***Name* :-** Abrar Mahedavi

***Roll No.* :-** 21035

***Title* :-** Extending to problem .2) Consider dictionary data is stored in a file in random order. Thus to search any word and its meaning from given data, program should create reasonably balance tree. Below program uses AVL tree for storing key and its meaning.

/\*

***Program :***

#include<iostream>

#include<string.h>

using namespace std;

class dictionary

{

public:

char key[20];

char meaning[20];

};

class node

{

public:

dictionary data;

node \*left,\*right;

int ht;

};

class AVL

{

node \*AVLroot;

node \*insert1(node\*, dictionary);

void preorder1(node \*);

void inorder1(node \*);

node \*rotateright(node \*);

node \*rotateleft(node \*);

node \*RR(node\*);

node \*LL(node\*);

node \*LR(node\*);

node \*RL(node\*);

int BF(node \*);

int height(node \*T);

public:

AVL()

{

AVLroot=NULL;

}

void insert(dictionary x)

{

AVLroot=insert1(AVLroot,x);

}

void preorder()

{

preorder1(AVLroot);

}

void inorder()

{

inorder1(AVLroot);

}

void search(dictionary x);

};

void AVL:: search(dictionary x)

{

node \*T=AVLroot;

while(T!=NULL)

{

if(strcmp(x.key,T->data.key)==0)

{

cout<<"Meaning is::"<<T->data.meaning;

return;

}

if(strcmp(x.key,T->data.key)>0)

T=T->right;

else

T=T->left;

}

cout<<"Key not found!!!";

}

node \* AVL ::insert1(node \*T,dictionary x)

{

if(T==NULL)

{

T=new node;

T->data=x;

T->left=NULL;

T->right=NULL;

}

else

if(strcmp(x.key,T->data.key)>0) // if data is greater, then

insert to right

{

T->right=insert1(T->right,x);

if(BF(T)==-2)

if(strcmp(x.key,T->right->data.key)>0)

T=RR(T); // rotate right

else

T=RL(T);

}

else // if data is less, then insert into left

if(strcmp(x.key,T->data.key)<0)

{

T->left=insert1(T->left,x);

if(BF(T)==2)

if(strcmp(x.key,T->left->data.key)<0)

T=LL(T); // rotate left

else

T=LR(T);

}

T->ht=height(T);

return(T);

}

int AVL::height(node \*T)

{

int lh,rh;

if(T==NULL)

return 0;

if(T->left==NULL)

lh=0;

else

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

if(T->right==NULL)

rh=0;

else

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

if(lh>rh)

return(lh);

else

return(rh);

}

node \*AVL::rotateright(node \* x)

{

node \*y; // y as temp node to swap and rotate

y=x->left;

x->left=y->right;

y->right=x;

x->ht=height(x);

y->ht=height(y);

return (y);

}

node \*AVL::rotateleft(node \* x)

{

node \*y; // y as temp node to swap and rotate

y=x->right;

x->right=y->left;

y->left=x;

y->ht=height(y);

x->ht=height(x);

return (y);

}

node \*AVL::RR(node \*T)

{

T=rotateleft(T);

return (T);

}

node \*AVL::LL(node \*T)

{

T=rotateright(T);

return (T);

}

node \*AVL::LR(node \*T)

{

T->left=rotateleft(T->left);

T=rotateright(T);

return (T);

}

node \*AVL::RL(node \*T)

{

T->right=rotateright(T->right);

T=rotateleft(T);

return (T);

}

int AVL::BF(node \*T)

{

int lh,rh;

if(T==NULL)

{

return 0;

}

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);

}

void AVL::preorder1(node \*T)

{

if(T!=NULL)

{

cout<<"\n"<<T->data.key<<"\t"<<T->data.meaning;

preorder1(T->left);

preorder1(T->right);

}

}

void AVL::inorder1(node \*T)

{

if(T!=NULL)

{

inorder1(T->left);

cout<<"\n"<<T->data.key<<"\t"<<T->data.meaning;

inorder1(T->right);

}

}

int main()

{

AVL A;

int n,i,op;

dictionary x;

do

{

cout<<"\n 1. Create";

cout<<"\n 2. Search";

cout<<"\n 3. Print";

cout<<"\n 4. Quit";

cout<<"\nEnter ur choice";

cin>>op;

switch(op)

{

case 1:

cout<<"Enter key and meaning";

cin>>x.key;

cin>>x.meaning;

A.insert(x);

break;

case 2:

cout<<"\n Enter data to search";

cin>>x.key;

A.search(x);

break;

case 3:

cout<<"\n Preorder is::";

A.preorder();

cout<<"\n Inorder is::";

A.inorder();

break;

}

}while(op!=4);

return 0;

}

***Output:***

compeng-sl2-08@compeng-sl2-08:~$ cd Abrar

compeng-sl2-08@compeng-sl2-08:~/Abrar$ g++ dict2.cpp

compeng-sl2-08@compeng-sl2-08:~/Abrar$ ./a.out

1. Create

2. Search

3. Print

4. Quit

Enter ur choice 1

Enter key and meaning mad

crazy

1. Create

2. Search

3. Print

4. Quit

Enter ur choice 1

Enter key and meaning banana

fruit

1. Create

2. Search

3. Print

4. Quit

Enter ur choice 1

Enter key and meaning king

ruler

1. Create

2. Search

3. Print

4. Quit

Enter ur choice 1

Enter key and meaning rose

flower

1. Create

2. Search

3. Print

4. Quit

Enter ur choice 3

**Preorder is ::**

king ruler

banana fruit

mad crazy

rose flower

**Inorder is::**

banana fruit

king ruler

mad crazy

rose flower

1. Create

2. Search

3. Print

4. Quit

Enter ur choice 2

Enter data to search **king**

Meaning is::**ruler**

1. Create

2. Search

3. Print

4. Quit

Enter ur choice 4