#include <iostream>

Using namespace std;

Class node

{ public:

String key;

String meaning;

Node \*left;

Node \*right;

};

Class Avl

{ node \*root;

Public:

Avl()

{root=NULL;

}

Void create():

Node\* insert(node \*curr,node \*temp);

Node\* balance(node \*temp);

Int diff(node \*temp);

Int height(node \*temp);

Int maximum(node \*temp);

Node\* ll(node \*temp);

Node\* rr(node \*temp);

Node\* lr(node \*temp);

Node\* rl(node \*temp);

Void ascending(node \*temp);

Node\* delete(node \*root,string key1);

Void deleten();

Void ascending(node \*temp);

Node\* delete\_n(node \*root,string key1);

Void deleten();

Node\* extractMin(node \*t);

Void descending(node \*temp);

Void display();

Bool search(node \*curr,string key1);

Void search\_value();

};

Void Avl::create()

{node \*temp;

Char ans;

Do

{temp=new node();

Cout<<”enter key value”;

Cin>>temp->key;

Cout<<”enter meaning of key”;

Cin>>temp->meaning;

Temp->left=temp->right-NULL;

Root=insert(root,temp);

Cout<<”do you want to add another node?(y/n)”;

Cin>>ans;

}while(ans==’y’||ans==’y’);

}

Node\* Avl ::insert(node \*curr,node \*temp)

{if(curr==NULL)

{return temp;}

If(temp->key<curr->key)

{curr->left=insert(curr->left,temp);

Curr=balance(curr);

}else if(temp->key>curr->key)

{curr->right=insert(curr->right,temp);

Curr=balance(curr);

}

Return curr;

}

Node\* Avl::balance(current \*temp)

{int bal;

Bal=diff(temp);

If(bal>=2)

{if(diff(temp->left)<0)

Temp=lr(temp);

Else

Temp=ll(temp);

}

Else if(bal<=-2)

{if(diff(temp->right)<0)

Temp=rl(temp);

Else

Temp=rr(temp);

}return temp;

}

Int Avl::diff(node \*temp)

{int l,r;

L=height(temp->left);

R=height(temp->right);

Return(l-r);

}

Int Avl::height(node \*temp)

{if(temp==NULL)

Return(-1);

Else

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

}

Int Avl::maximum(int a,int b)

{if(a>b)

Return a;

Else

Return b;

}

Node Avl::ll(node \*par)

{node \*temp,\*temp1;

Temp=par->left;

Temp1=temp->right;

Temp->right=par;

Par->left=temp1;

Return temp;

}

Node Avl::rr(node \*par)

{node \*temp,\*temp1;

Temp=par->right;

Temp1=temp->left;

Temp->left=par;

Par->right=temp1;

Return temp;

}

Node\* Avl::rl(node \*par)

{par->right=ll(par->right);

Return(rr(par));

}

Node\* Avl::lr(node \*par)

{par->left=rr(pa->left);

Return(ll(par));

}

Void Avl::ascending(node \*temp)

{if(temp!=NULL)

{ascending(temp->left);

Cout<<”\n\t”<<temp->key<<”\n\t”<<temp->meaning;

Ascending(temp->right);

}

}

Void Avl::descending(node \*temp)

{if(temp!=NULL)

{descending(temp->right);

Cout<<”\n\t”<<temp->key<<”\n\t”<<temp->meaning;

Descending(temp->left);

}

}

Void AVL::display()

{cout<<”the ascending order are:”;

Ascending(root);

Cout<<”the descending order are:”;

Descending(root);

}

Bool AVL::search( node \*cur,string key1)

{if(curr)

{

If(curr->key==key1)

Return true;

If(curr->key>key1)

Return search( curr->left,key1);

Else

Return search(curr->right,key1);

}

Return false;

}

Void AVL::search\_value()

{

String key2;

Cout<<”\n Enter the keyword you wish to search”;

Cin>>key2;

If(search(root,key2))

Cout<<”\n The entered keyword isin the AVl tree”;

Else

Cout<<”\n The entered keyword is not resent in the AVL tree”;

}

Node\* AVL: :delete\_n( node\* curr,string key1)

{

If ( !curr)

Return curr;

If ( keyl < curr->key)

Curr->left=delete\_n(curr->left,key1);

Else if( keyl>curr->key)

Curr->right=delete\_n(curr->right,key1);

Else

{

Node \*l=curr->left;

Node \*r=curr->right;

Delete curr;

If(!r)

Return 1;

Node \*M=r;

While(M->left )

M=M->left:

M->right = extractMin®:

M->left = 1;

Return balance(M):

}

Return balance( curr);

}

Node\* AVL::extractMin(node \*t)

{

If ( !t->left)

Return t->right;

t->left = extractMin(t->left);

return balance(t);

}

Void AVL::deleten()

{

String key:

Cout<< “\n Enter the keyword to be deleted ： “;

Cin>>key;

Root=delete\_n(root,key);

}

Int main()

{

Char c;

Int ch;

AVL a;

Do

{

Cout<<”\n l.Insert a keyword in AVL tree.”;

Cout<<”\n 2.Display the AVL tree.”;

Cout<<”\n 3.Search a keyword”;Cout<<”\n 4.Delete a keyword.”;cout<<”\n Enter your choice ： “;

Cin>>ch;

Switch(ch)

{case 1: a.create();

Break;

Case 2: a.display();

Break;

Case 3: a.search\_value();

Break;

Case 4: a.deleten();

Break;

Default:cout<<”wrong choice”;

}

Cout<<”do u want to continue?(y/n)”;

Cin>>c;

}

While(c==’y’||c==’Y’);

Return 0;

}