**PROGRAM FOR BINERY SEARCH TREE(DICTIONARY):**

#include<iostream>

#include<cstdlib>

#include<cstring>

using namespace std;

class node

{

public:

char word[20],mean[50];

node \*left;

node \*right;

};

class BST

{

private:

node \*root;

public:

BST()

{

root=NULL;

return;

}

void insert(char word[20],char mean[50]);

void create();

void search();

void display(node\* node);

node \*delet(node \*head,char word[20]);

node \*In\_succ(node \*trav);

void ascending(node \*node);

void descending(node \*node);

void traversal(node \*node);

void construction();

void update();

void max\_comp();

};

void BST::insert(char word[20],char mean[50])

{

node \*temp;

temp=new node();

strcpy(temp->word,word);

strcpy(temp->mean,mean);

if(root==NULL)

{

root=temp;

}

else

{

node \*curr,\*parent;

curr=root;

while(curr!=NULL)

{

parent=curr;

if(strcmp(word,curr->word)<0)

{

curr=curr->left;

}

else

{

curr=curr->right;

}

if(strcmp(word,parent->word)<0&&parent->left==NULL)

{

parent->left=temp;

}

if(strcmp(word,parent->word)>0&&parent->right==NULL)

{

parent->right=temp;

}

}

}

}

void BST::create()

{

int n,i;

char word[20],mean[50];

cout<<"Enter number of words\n";

cin>>n;

cout<<"Enter word and its meaning\n";

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

{

cin>>word>>mean;

insert(word,mean);

}

}

void BST::search()

{

int flag=0;

char word[20];

cout<<"Enter word to be searched\n";

cin>>word;

node \*trav;

trav=root;

while(trav!=NULL)

{

if(strcmp(word,trav->word)==0)

{

flag=1;

break;

}

if(strcmp(word,trav->word)<0)

{

trav=trav->left;

}

else

{

trav=trav->right;

}

}

if(flag==1)

{

cout<<"Word found\n";

}

else

cout<<"Word not found\n";

}

void BST::display(node\* head)

{

if(head==NULL)

return;

cout<<endl<<head->word<<" - "<<head->mean;

display(head->left);

display(head->right);

}

node \*BST::delet(node \*head,char word[20])

{

node \*temp;

if(head==NULL)

{

return NULL;

}

if(strcmp(word,head->word)>0)

{

head->right=delet(head->right,word);

return head;

}

if(strcmp(word,head->word)<0)

{

head->left=delet(head->left,word);

return head;

}

if(head->left==NULL&&head->right==NULL)

{

temp=head;

head=NULL;

delete temp;

return head;

}

if(head->left==NULL)

{

temp=head;

head=head->right;

delete temp;

return head;

}

if(head->right==NULL)

{

temp=head;

head=head->left;

delete temp;

return head;

}

temp=In\_succ(head->right);

strcpy(head->word,temp->word);

strcpy(head->mean,temp->mean);

head->right=delet(head->right,temp->word);

return head;

}

node \*BST::In\_succ(node \*x)

{

node \*trav;

trav=x;

while(trav->left!=NULL)

{

trav=trav->left;

}

return trav;

}

void BST::ascending(node\* head)

{

if(head!=NULL)

{

ascending(head->left);

cout<<endl<<head->word<<" - "<<head->mean;

ascending(head->right);

}

}

void BST::descending(node\* head)

{

if(head==NULL)

return;

descending(head->right);

cout<<endl<<head->word<<" - "<<head->mean;

descending(head->left);

}

void BST::traversal(node \*head)

{

cout<<"Pre order traversal\n";

display(head);

cout<<"\nIn order traversal\n";

ascending(head);

cout<<"\nPost order traversal\n";

if(head==NULL)

return;

display(head->left);

display(head->right);

cout<<endl<<head->word<<" - "<<head->mean;

}

void BST::update()

{

int flag=0;

char word[20],new\_word[20],mean[50];

cout<<"Enter word to be updated\n";

cin>>word;

cout<<"Enter new word\n";

cin>>new\_word;

cout<<"Enter updated meaning\n";

cin>>mean;

node \*trav;

trav=root;

while(trav!=NULL)

{

if(strcmp(word,trav->word)==0)

{

flag=1;

strcpy(trav->word,new\_word);

strcpy(trav->mean,mean);

break;

}

if(strcmp(word,trav->word)<0)

{

trav=trav->left;

}

else

{

trav=trav->right;

}

}

if(flag==1)

{

cout<<"Word updated successfully\n";

}

else

cout<<"Word not found\n";

}

void BST::max\_comp()

{

int flag=0,count=1;

char word[20];

cout<<"Enter word to be searched\n";

cin>>word;

node \*trav;

trav=root;

while(trav!=NULL)

{

if(strcmp(word,trav->word)==0)

{

flag=1;

break;

}

if(strcmp(word,trav->word)<0)

{

trav=trav->left;

}

else

{

trav=trav->right;

}

count++;

}

if(flag==1)

{

cout<<"Max comparisons needed to search given word\n";

cout<<count;

}

else

cout<<"Word not found\n";

}

void BST::construction()

{

int ch;

char word[20],mean[50];

create();

do

{

cout<<"\n------------------\n";

cout<<"Menu\n1.Insert a word\n2.Delete a word

\n3.Display dictionary\n4.Search a word\n";

cout<<"5.Ascending order\n6.Descending order\n7.Traversal\n8.Update a word\n";

cout<<"9.Max comparisons\n10.Exit\n";

cout<<"------------------\n";

cout<<"Enter your choice\n";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter a word to insert\n";

cin>>word;

cout<<"Enter its meaning\n";

cin>>mean;

insert(word,mean);

break;

case 2:

node \*head;

head=root;

cout<<"Enter a word to delete\n";

cin>>word;

root=delet(head,word);

cout<<"Word deleted successfully\n";

break;

case 3:

cout<<"Dictionary \n";

display(root);

break;

case 4:

search();

break;

case 5:

cout<<"Ascending order\n";

ascending(root);

break;

case 6:

cout<<"Descending order\n";

descending(root);

break;

case 7:

traversal(root);

break;

case 8:

update();

break;

case 9:

max\_comp();

break;

case 10:

exit(0);

default:

cout<<"Invalid choice\n";

break;

}

}while(ch!=10);

}

int main()

{

BST bst;

bst.construction();

return 0;

}

**OUTPUT**:

Enter number of words

5

Enter word and its meaning

a apple

b ball

c cat

d dog

e egg

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

1

Enter a word to insert

f

Enter its meaning

frog

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

3

Dictionary

a - apple

b - ball

c - cat

d - dog

e - egg

f - frog

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

2

Enter a word to delete

a

Word deleted successfully

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

3

Dictionary

b - ball

c - cat

d - dog

e - egg

f - frog

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

4

Enter word to be searched

b

Word found

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

5

Ascending order

b - ball

c - cat

d - dog

e - egg

f - frog

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

6

Descending order

f - frog

e - egg

d - dog

c - cat

b - ball

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

7

Pre order traversal

b - ball

c - cat

d - dog

e - egg

f - frog

In order traversal

b - ball

c - cat

d - dog

e - egg

f - frog

Post order traversal

c - cat

d - dog

e - egg

f - frog

b - ball

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

8

Enter word to be updated

f

Enter new word

h

Enter updated meaning

horse

Word updated successfully

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

3

Dictionary

b - ball

c - cat

d - dog

e - egg

h - horse

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

9

Enter word to be searched

d

Max comparisons needed to search given word

3

------------------

Menu

1.Insert a word

2.Delete a word

3.Display dictionary

4.Search a word

5.Ascending order

6.Descending order

7.Traversal

8.Update a word

9.Max comparisons

10.Exit

------------------

Enter your choice

10