Exp 2

**INPUT::**

Write C++ program to implement Dictionary using binary search tree.

#include <iostream>

#include<string>

using namespace std;

class dictionary;

class node

{

 string word,meaning;

 node \*left,\*right;

public:

 friend class dictionary;

 node()

 {

  left=NULL;

  right=NULL;

 }

 node(string word, string meaning)

 {

  this->word=word;

  this->meaning=meaning;

  left=NULL;

  right=NULL;

 }

};

class dictionary

{

public:

  node \*root;

 dictionary()

{

  root=NULL;

}

 void create();

 void inorder\_rec(node \*rnode);

 void descending\_rec(node \*rnode);

 void inorder()

 {

  inorder\_rec(root);

 }

 void descending();

 bool insert(string word,string meaning);

 int search(string key);

 node\* deleteNode(node \* root1, string key);

 node\*  successor(node \* root1);

node\*  predecessor(node \* root1);

};

int dictionary::search(string key)

{

 node \*tmp=root;

 int count;

 if(tmp==NULL)

 {

  return -1;

 }

 if(root->word==key)

  return 1;

 while(tmp!=NULL)

 {

  if((tmp->word)>key)

  {

   tmp=tmp->left;

   count++;

  }

  else if((tmp->word)<key)

  {

   tmp=tmp->right;

   count++;

  }

  else if(tmp->word==key)

  {

   return ++count;

  }

 }

 return -1;

}

void dictionary::descending()

{

 descending\_rec(root);

}

void dictionary::descending\_rec(node \*rnode)

{

 if(rnode)

 {

  descending\_rec(rnode->right);

  cout<<" "<<rnode->word<<" : "<<rnode->meaning<<endl;

        descending\_rec(rnode->left);

  }

}

void dictionary::create()

{

 int n;

 string wordI,meaningI;

 cout<<"\nHow many Word to insert?:\n";

 cin>>n;

 for(int i=0;i<n;i++)

 {

  cout<<"\nENter Word: ";

  cin>>wordI;

  cout<<"\nEnter Meaning: ";

  cin>>meaningI;

  insert(wordI,meaningI);

 }

}

void dictionary::inorder\_rec(node \*rnode)

{

 if(rnode)

 {

  inorder\_rec(rnode->left);

  cout<<" "<<rnode->word<<" : "<<rnode->meaning<<endl;

  inorder\_rec(rnode->right);

 }

}

bool dictionary::insert(string word, string meaning)

{

 node \*p=new node(word, meaning);

 if(root==NULL)

 {

  root=p;

  return true;

 }

 node \*cur=root;

 node \*par=root;

 while(cur!=NULL) //traversal

 {

  if(word>cur->word)

  {par=cur;

  cur=cur->right;

  }

  else if(word<cur->word)

  {

   par=cur;

   cur=cur->left;

  }

  else

  {

   cout<<"\nWord is already in the dictionary.";

   return false;

  }

 }

 if(word>par->word) //insertion of node

 {

  par->right=p;

  return true;

 }

 else

 {

  par->left=p;

  return true;

 }

}

node\* dictionary:: successor(node \* root1) {

  root1 = root1 -> right;

  while (root1 -> left != NULL) root1 = root1 -> left;

  return root1;

}

node\* dictionary:: predecessor(node \* root1) {

  root = root1 -> left;

  while (root1 -> right != NULL) root1 = root1 -> right;

  return root1;

}

node \* dictionary:: deleteNode(node \* root1, string key)

{

  node \*temp;

  if (root1 == NULL) return NULL;

  if (key > root1 -> word) root1 -> right = deleteNode(root1 -> right, key);

  else if (key < root1 -> word) root1->left = deleteNode(root1 -> left, key);

  else {

    if (root1 -> left == NULL && root1 -> right == NULL) root1 = NULL;

    else if (root1 -> right != NULL) {

      temp= successor(root1);

      root1->word=temp->word;

      root1->meaning=temp->meaning;

      root1->right = deleteNode(root1 -> right, root1 -> word);

    } else {

      temp = predecessor(root1);

      root1->word=temp->word;

      root1->meaning=temp->meaning;

      root1 -> left = deleteNode(root1 -> left, root1 -> word);

    }

  }

  return root1;

}

int main() {

 string word;

 dictionary months;

 months.create();

 cout<<"Ascending order\n";

 months.inorder();

 cout<<"\nDescending order:\n";

 months.descending();

 cout<<"\nEnter word to search: ";

 cin>>word;

 int comparisons=months.search(word);

 if(comparisons==-1)

 {

  cout<<"\nNot found word";

 }

 else

 {

  cout<<"\n "<<word<<" found in "<<comparisons<<" comparisons";

 }

 cout<<"\nEnter word to be delete: ";

 cin>>word;

 months.root= months.deleteNode(months.root,word);

 months.inorder();

 return 0;

}

**Output::**

How many Word to insert?:

5

ENter Word: car

Enter Meaning: vehicle

ENter Word: chips

Enter Meaning: eatery

ENter Word: pants

Enter Meaning: cloth

ENter Word: girl

Enter Meaning: female

ENter Word: boy

Enter Meaning: male

Ascending order

boy : male

car : vehicle

chips : eatery

girl : female

pants : cloth

Descending order:

pants : cloth

girl : female

chips : eatery

car : vehicle

boy : male

Enter word to search: girl

girl found in 4 comparisons

Enter word to be delete: boy

car : vehicle

chips : eatery

girl : female

pants : cloth