PROJECT REPORT

OF

DATA STRUCTURES

ON

DICTIONARY WHICH STORES KEYWORDS AND MEANINGS

(USING BINARY SEARCH TREE)

SUBMITTED BY:

Shubham Singh	9920103111
Vanshita Varshney	9920103107
Ananya Aggarwal	9920103107
Chandrika Rajvanshi	9920103096

Department of CSE/IT

Jaypee Institute of Information Technology University, Noida

December, 2022

INTRODUCTION

A Dictionary stores keywords & its meanings, provides facility for adding new keywords, deleting keywords, updating values of any entry. It also provides facility to display whole data sorted in ascending/ Descending order. It can be used to find how many maximum comparisons may be required for finding any keyword.

APPLICATIONS

Class dictionary () is our main function

Class node () is used to pass arguments of meaning and word

We have used pointers root, left and right

Create () is function used to take input from user

Post order () is function used to print values in descending order

In order () is function used to print values in ascending order

Search () is function used to search the values inputted by the user

SOFTWARE REQUIREMENTS

Online C++ compiler

HARDWARE REQUIREMENTS

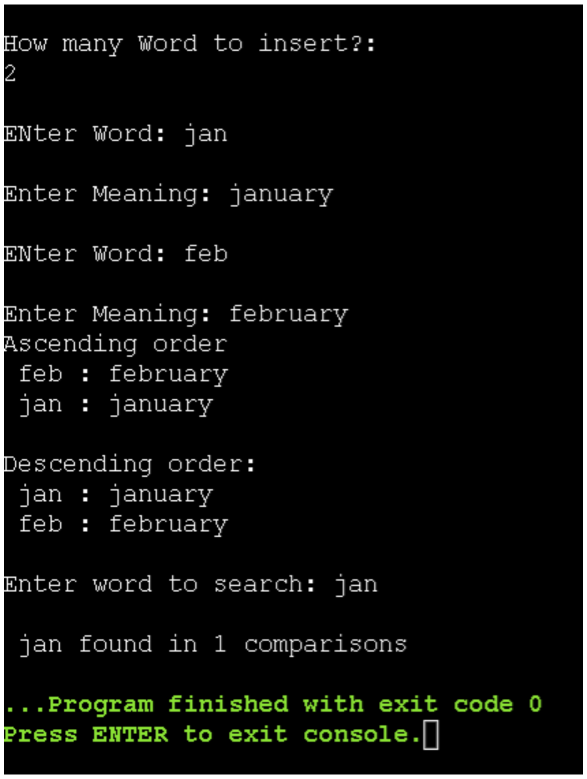
Laptop

CODE

```
//============================================================================
// Name        : DictionaryBST.cpp
// Author      : VNKDJ5
//============================================================================

#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
{
 node *root;
public:
 dictionary()
{
  root=NULL;
}
 void create();
 void inorder_rec(node *rnode);
 void postorder_rec(node *rnode);
 void inorder()
 {
  inorder_rec(root);
 }
 void postorder();

 bool insert(string word,string meaning);
 int search(string key);

};
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::postorder()
{
 postorder_rec(root);
}
void dictionary::postorder_rec(node *rnode)
{
 if(rnode)
 {
  postorder_rec(rnode->right);
  cout<<" "<<rnode->word<<" : "<<rnode->meaning<<endl;
  postorder_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;
 }
}

int main() {
 string word;
 dictionary months;
 months.create();
 cout<<"Ascending order\n";
 months.inorder();

 cout<<"\nDescending order:\n";
 months.postorder();

 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";
 }
 return 0;
}
```

OUTPUT



```
How many Word to insert?:
2

ENter Word: jan

Enter Meaning: january

ENter Word: feb

Enter Meaning: february
Ascending order
 feb : february
 jan : january

Descending order:
 jan : january
 feb : february

Enter word to search: jan

 jan found in 1 comparisons

...Program finished with exit code 0
Press ENTER to exit console.
```