Name:- Amit Bandu Swami

Roll No. :- 2221018

Class :- SE COMP

Ass 2

Problem :- A Dictionary stores keywords and its meanings. Provide facility for adding new keywords, deleting keywords, updating values of any entry. Provide facility to display whole data sorted in ascending/ Descending order. Also find how many maximum comparisons may require for finding any keyword. Use Binary Search Tree for implementation.

```
#include<iostream>
#include<string.h>
using namespace std;
class node
{
    public:
      char word[50],mean[50];
      node *l,*r;
};
class dictionary
{
    public:
      void create(node *root);
      void inorder(node *root);
      void insert(node *root);
      node *del(node *root,char key[]);
      void update(node *root);
      node *smallest(node *root);
};
void dictionary::create(node *root)
{
    int i,ans;
    node *ne,*temp;
    do
    {
        ne=new node;
        cout<<"Enter new node word:"<<endl;
        cin>>ne->word;
        cout<<"Enter new node word meanig:"<<endl;
        cin>>ne->mean;
        ne->r=NULL;
        ne->l=NULL;
        temp=root;
        while(1)
        {
            i=strcmp(ne->word,temp->word);
            if(i<0)
            {
```

```
            if(temp->l==NULL)
            {
                temp->l=ne;
                break;
            }
            temp=temp->l;
        }
        else if(i>0)
        {
            if(temp->r==NULL)
            {
                temp->r=ne;
                break;
            }
            temp=temp->r;
        }
        else
        {
            cout<<"You added Duplicate word!!"<<endl;
            break;
        }
    }
    cout<<"Do you want to add new node(1/0):"<<endl;
    cin>>ans;
}while(ans==1);
}
void dictionary::inorder(node *root)
{
    if(root!=NULL)
    {
        inorder(root->l);
        cout<<root->word<<" "<<root->mean<<endl;
        inorder(root->r);
    }

}
void dictionary::insert(node *root)
{
    node *ne,*temp;
    int i;
    ne=new node;
    cout<<"Enter new node word:"<<endl;
    cin>>ne->word;
    cout<<"Enter new node word meanig:"<<endl;
    cin>>ne->mean;
    ne->r=NULL;
    ne->l=NULL;
    temp=root;
    while(1)
    {
    i=strcmp(ne->word,temp->word);
    if(i<0)
    {
        if(temp->l==NULL)
        {
            temp->l=ne;
            break;
        }
        temp=temp->l;
    }
    else if(i>0)
    {
        if(temp->r==NULL)
        {
                temp->r=ne;
            break;
        }
        temp=temp->r;
    }
    else
    {
        cout<<"You added Duplicate word!!"<<endl;
        break;
    }
    }
}
node* dictionary::smallest(node *root)
{
    node *temp;
    temp=root;
    while(temp->l!=NULL)
    {
    temp=temp->l;
    }
    return temp;
}
node* dictionary::del(node *root,char key[])
{
    node *small;
    int i;
    if(root==NULL)
    return root;
    i=strcmp(key,root->word);
    if(i<0)
    {
    root->l=del(root->l,key);
    }
    else if(i>0)
    {
    root->r=del(root->r,key);
    }
    else
    {
    if(root->r!=NULL)
    {
        small=smallest(root->r);
        strcpy(root->word,small->word);
        strcpy(root->mean,small->mean);
        root->r=del(root->r,small->word);
    }
    else
    {
        return root->l;
    }
    }
    return root;
}

void dictionary::update(node *root)
{
  node *temp;
  int i;
  char key[20];
  cout<<"Enter the word whose meaning you want to update:"<<endl;
  cin>>key;
  temp=root;
  while(temp!=NULL)
  {
    i=strcmp(key,temp->word);
    if(i==0)
    {
      cout<<"The word found!!"<<endl;
      cout<<"Enter the new meaning:"<<endl;
      cin>>temp->mean;
      break;
    }
    else if(i<0)
    {
      temp=temp->l;
    }
    else
    {
      temp=temp->r;
    }
  }
  if(temp==NULL)
  {
    cout<<"Word not found!!\n";
  }
}
```

```
int main()
{
   dictionary  ob;
   node *root,*d;
   int ch;
   char key[10];
   while(1)
   {
      cout<<"1. Create"<<endl;
      cout<<"2. Inorder"<<endl;
      cout<<"3.  Insert"<<endl;
      cout<<"4. Delete"<<endl;
      cout<<"5. Update"<<endl;
      cout<<"Enter your choice:"<<endl;
      cin>>ch;
      switch(ch)
      {
         case 1:root=new  node;
              cout<<"Enter root word:"<<endl;
              cin>>root->word;
              cout<<"Enter root word Meaning:"<<endl;
              cin>>root->mean;
              root->l=NULL;
              root->r=NULL;
              ob.create(root);
              break;
         case 2:ob.inorder(root);
              break;
         case 3:ob.insert(root);
              break;
         case 4:cout<<"Enter key to delete:"<<endl;
              cin>>key;
              root=ob.del(root,key);
              cout<<"node is deleted"<<endl;
              ob.inorder(root);
              break;
         case 5:ob.update(root);
              break;
      }
   }
}
```