PYTHON PROGRAM FOR IMPLEMENTING WORD DICTIOANRY USING BINARY SEARCH TREE.

PROGRAM:

```
"""FOLLOWING PROGRAM IMPLEMENTS WORD DICTIONARY USING BST"""

class Node:
    """Node class for structure of each word""" def __init__(self,word,mean):
        self.left=None
        self.word=word
        self.mean=mean
        self.right=None

        """insertion opeartion""" def insert(self,word,mean): if self.word==word:
            return False elif self.word>word:
            if self.left:
    return self.left.insert(word,mean) else:

            self.left=Node(word,mean)

        return True else: if self.right:
    return self.right.insert(word,mean) else:
```

```
                self.right=Node(word,mean)
                    return True

        """seraching opeartion""" def
        search(self,word):
                if(self.word==word): return
                                self.mean elif
                            self.word>word:
                    if self.left:

        return self.left.search(word) else: return False else: if

                self.right:

        return self.right.search(word) else: return False

        """Traversing""" def
        inorder(self):
                if self:
                    if self.left:
        self.left.inorder() print(str(self.word)," :",str(self.mean)) if self.right:
                            self.right.inorder()



class Tree:
        """implementation of tree""" def
        __init__(self):
                self.root=None

        """insertion opeartion""" def
```

```
insert(self,word,mean):
        if self.root:
return self.root.insert(word,mean) else:

        self.root=Node(word,mean)
                return True

"""search opearation""" def
search(self,word):
        if self.root:

return self.root.search(word) else: return False

"""traversing""" def
inorder(self):
        self.root.inorder()

"""removal of node""" def
remove(self, data):
        if self.root is None:
print("Dictionary is empty") elif self.root.word ==
        data:
                if self.root.left is None and self.root.right is None:
self.root = None elif self.root.left and self.root.right is None:
self.root = self.root.left elif self.root.left is None and self.root.right:
                        self.root = self.root.right elif self.root.left and
                                        self.root.right:
                        delNodeParent = self.root delNode =
                        self.root.right while delNode.left:
                                delNodeParent = delNode delNode =
                                        delNode.left self.root.word =
```

```
                                delNode.word if delNode.right:
                            if delNodeParent.word > delNode.word:
                        delNodeParent.left = delNode.right elif delNodeParent.word
                        < delNode.word: delNodeParent.right = delNode.right else:
                            if delNode.word < delNodeParent.word:

delNodeParent.left = None else: delNodeParent.right = None
        return parent = None node = self.root

        while node and node.word != data:
                parent = node if data <
                node.word: node = node.left elif
                data > node.word:
                        node = node.right

        if node is None or node.word != data:
                print("oops! word in not in dictionary")

        elif node.left is None and node.right is None:
                if data < parent.word: parent.left =

                    None else: parent.right = None
                                        return True

        elif node.left and node.right is None:
                if data < parent.word:

parent.left = node.left else: parent.right = node.left
                return True

        elif node.left is None and node.right:
                if data < parent.word:
```

```
parent.left = node.right else: parent.right = node.right
                return True


            else:    delNodeParent    =    node
            delNode    =    node.right    while
            delNode.left:
                    delNodeParent = delNode delNode =
                                delNode.left node.word =
                            delNode.word if delNode.right:
                        if delNodeParent.word > delNode.word:
                    delNodeParent.left = delNode.right elif delNodeParent.word
                < delNode.word: delNodeParent.right = delNode.right else: if
                delNode.word < delNodeParent.word: delNodeParent.left =
                            None else: delNodeParent.right = None



"""creation of tree"""

tree=Tree()

print("----------------------WORD DICTIONARY---------------- ----------") while 1:

"""taking choice from user""" choice=input("enter the choice : 1. ADD
2.SEARCH 3.DELETE 4.DISPLAY ....")
    if choice=='1':

        """insertion""" print("----------------------") word=input("enter the word to
        add to dictionary : ") mean=input("enter the meaning of the data : ")
```

```
            tree.insert(word,mean)

        elif choice=='2':

"""searching""" print("----------------------") word=input("enter the word
to serach in the dictionary : ")
            ser=tree.search(word) if ser is
            not False:
             print("----------------------") print("Meaning of ",word," is :
                    ",ser) print("----------------------") else:

                            print("----------------------")
                    print("sorry ! word not found")
                    print("----------------------")

        elif choice=='3':

"""deleting""" print("----------------------") word=input("Enter the
word to delete from dictionary: ")
            tree.remove(word) print("----------------------")

        elif choice=='4':

"""printing tree""" print("----------------------Tree is---------------- ---------")
        tree.inorder() print("----------------------") else:

        print("wrong choice")
```