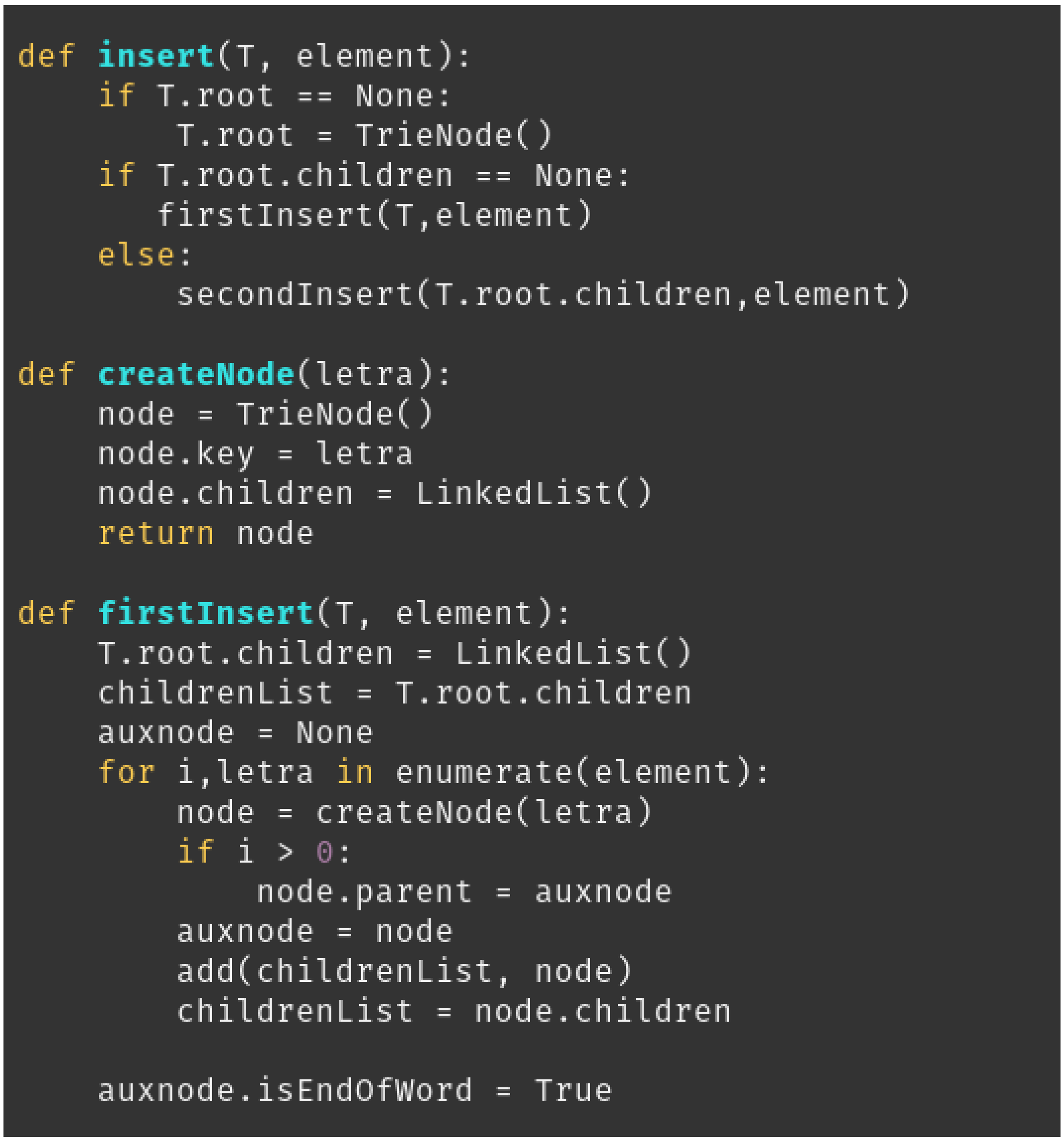
Parte 1

```
class Trie:
     root = None

class TrieNode:
     parent = None
     children = None
     key = None
     isEndOfWord = False
```

Ejercicio 1

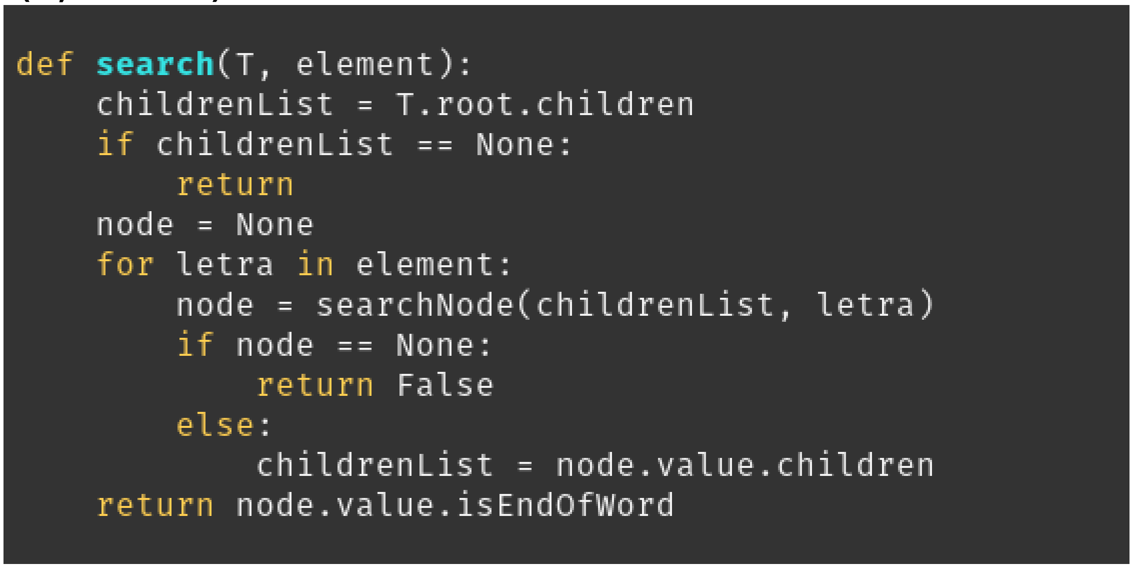
`insert(T,element)`



```
def insert(T, element):
    if T.root == None:
        T.root = TrieNode()
    if T.root.children == None:
       firstInsert(T,element)
    else:
        secondInsert(T.root.children,element)

def createNode(letra):
    node = TrieNode()
    node.key = letra
    node.children = LinkedList()
    return node

def firstInsert(T, element):
    T.root.children = LinkedList()
    childrenList = T.root.children
    auxnode = None
    for i,letra in enumerate(element):
        node = createNode(letra)
        if i > 0:
            node.parent = auxnode
        auxnode = node
        add(childrenList, node)
        childrenList = node.children

    auxnode.isEndOfWord = True
```

```python
def secondInsert(T, element):
    childrenList = T
    auxNode = None
    node = searchNode(childrenList, element[0])
    if node == None:
        for i,letra in enumerate(element):
            node = createNode(letra)
            if i > 0:
                node.parent = auxNode
            auxNode = node
            add(childrenList, node)
            childrenList = node.children

        auxNode.isEndOfWord = True
    else:
        auxNode = node.value
        for i, letra in enumerate(element):
            if i > 0:
                node = searchNode(childrenList, letra)
            if node != None:
                childrenList = node.value.children
                auxNode = node.value
            else:
                nodo = createNode(letra)
                if i > 0:
                    nodo.parent = auxNode
                auxNode = nodo
                add(childrenList, nodo)
                childrenList = nodo.children
        auxNode.isEndOfWord = True

def searchNode(L, element):
    if L.head == None:
        return
    currentNode = L.head
    while currentNode != None:
        if currentNode.value.key == element:
            return currentNode
        currentNode = currentNode.nextNode
```

`search(T,element)`



```python
def search(T, element):
    childrenList = T.root.children
    if childrenList == None:
        return
    node = None
    for letra in element:
        node = searchNode(childrenList, letra)
        if node == None:
            return False
        else:
            childrenList = node.value.children
    return node.value.isEndOfWord
```

Ejercicio 2

La solucion seria usar en vez de usar listas como estructura de datos para el children, usariamos Array, con todo el abecedario pre cargado

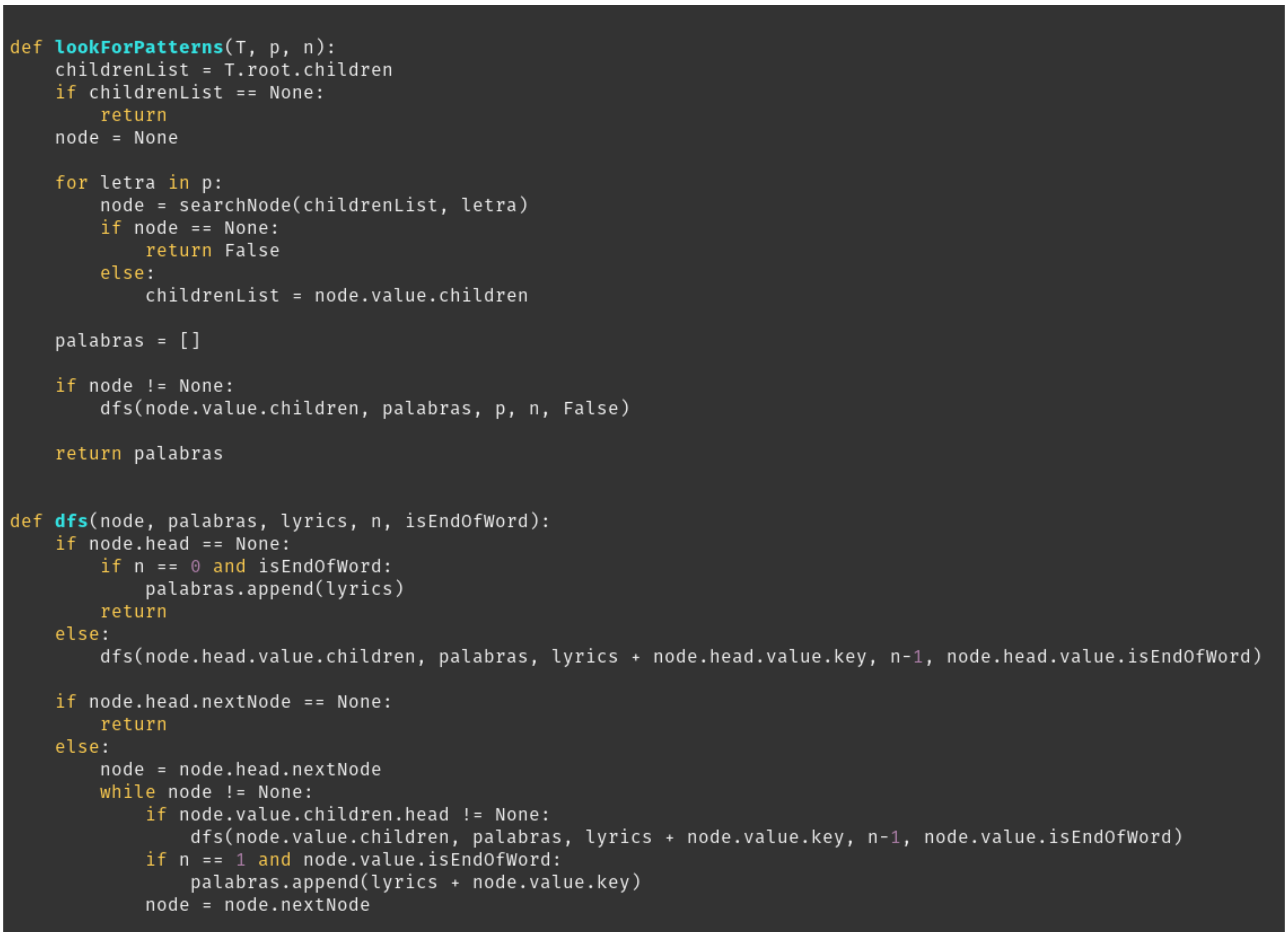
Ejercicio 3

`delete(T,element)`

```python
def delete(T, element):
    childrenList = T.root.children
    if childrenList == None:
        return

    node = None

    for letra in element:
        node = searchNode(childrenList, letra)
        if node == None:
            return False
        else:
            childrenList = node.value.children

    if node.value.isEndOfWord :
        if node.value.children.head == None:
            parent = node.value.parent
            letra = node.value.key
            node.value.parent = None
            linkedList.delete(parent.children, letra)
            deleteR(T,parent)
            return True
        else:
            node.value.isEndOfWord = False
    else:
        return False


def deleteR(T,node):
    if node.isEndOfWord or node == None:
        return
    else:
        if node.parent == None:
            linkedList.delete(T.root.children, node.key)
        else:
            parent = node.parent
            letra = node.key
            node.parent = None
            linkedList.delete(parent.children, letra)
            deleteR(T,parent)
```

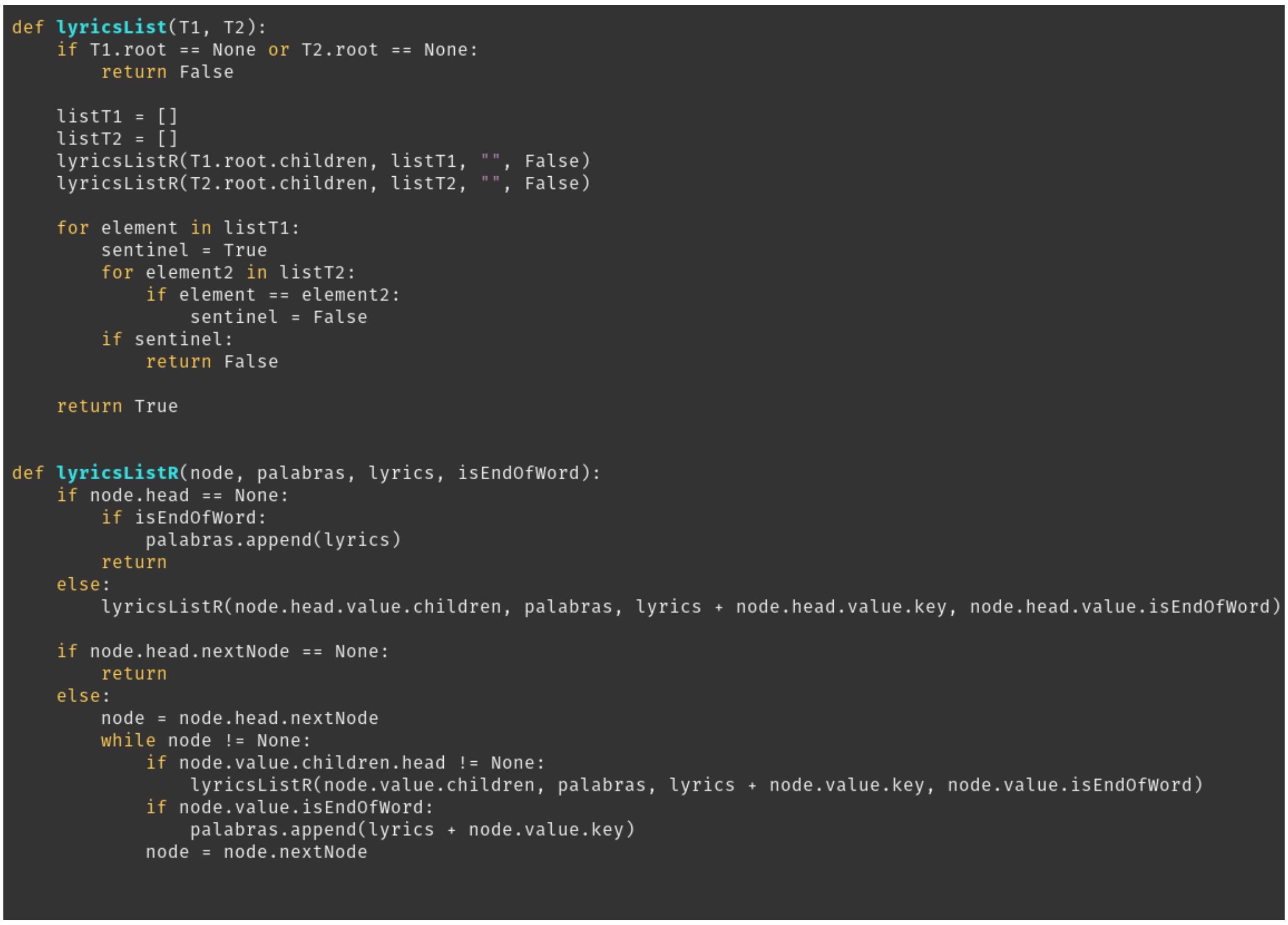
Parte 2

Ejercicio 4



```python
def lookForPatterns(T, p, n):
    childrenList = T.root.children
    if childrenList == None:
        return
    node = None

    for letra in p:
        node = searchNode(childrenList, letra)
        if node == None:
            return False
        else:
            childrenList = node.value.children

    palabras = []

    if node != None:
        dfs(node.value.children, palabras, p, n, False)

    return palabras


def dfs(node, palabras, lyrics, n, isEndOfWord):
    if node.head == None:
        if n == 0 and isEndOfWord:
            palabras.append(lyrics)
        return
    else:
        dfs(node.head.value.children, palabras, lyrics + node.head.value.key, n-1, node.head.value.isEndOfWord)

    if node.head.nextNode == None:
        return
    else:
        node = node.head.nextNode
        while node != None:
            if node.value.children.head != None:
                dfs(node.value.children, palabras, lyrics + node.value.key, n-1, node.value.isEndOfWord)
            if n == 1 and node.value.isEndOfWord:
                palabras.append(lyrics + node.value.key)
            node = node.nextNode
```

Ejercicio 5

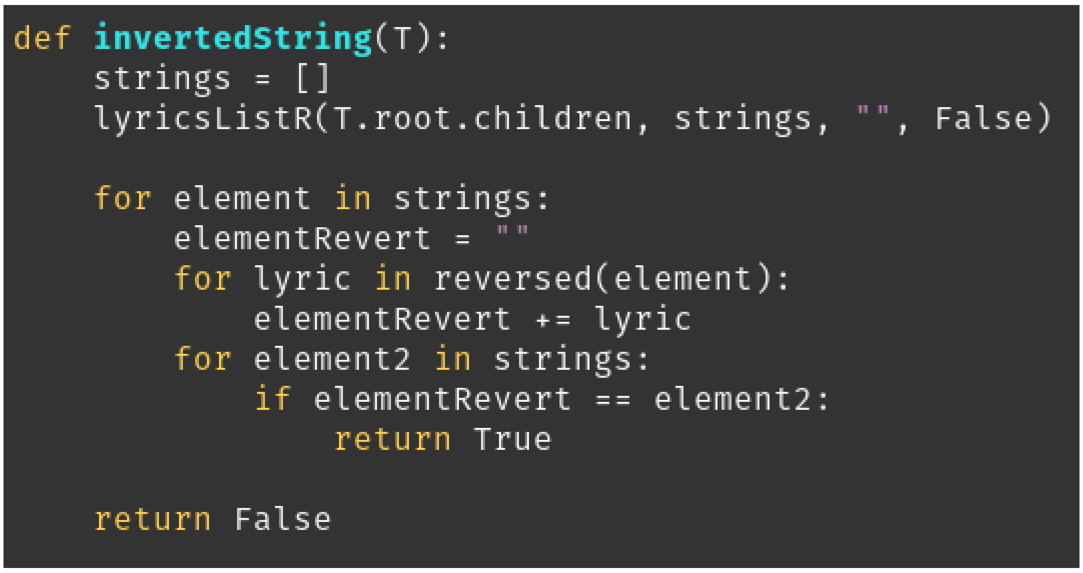


```python
def lyricsList(T1, T2):
    if T1.root == None or T2.root == None:
        return False

    listT1 = []
    listT2 = []
    lyricsListR(T1.root.children, listT1, "", False)
    lyricsListR(T2.root.children, listT2, "", False)

    for element in listT1:
        sentinel = True
        for element2 in listT2:
            if element == element2:
                sentinel = False
        if sentinel:
            return False

    return True


def lyricsListR(node, palabras, lyrics, isEndOfWord):
    if node.head == None:
        if isEndOfWord:
            palabras.append(lyrics)
        return
    else:
        lyricsListR(node.head.value.children, palabras, lyrics + node.head.value.key, node.head.value.isEndOfWord)

    if node.head.nextNode == None:
        return
    else:
        node = node.head.nextNode
        while node != None:
            if node.value.children.head != None:
                lyricsListR(node.value.children, palabras, lyrics + node.value.key, node.value.isEndOfWord)
            if node.value.isEndOfWord:
                palabras.append(lyrics + node.value.key)
            node = node.nextNode
```

Analizar el costo computacional:

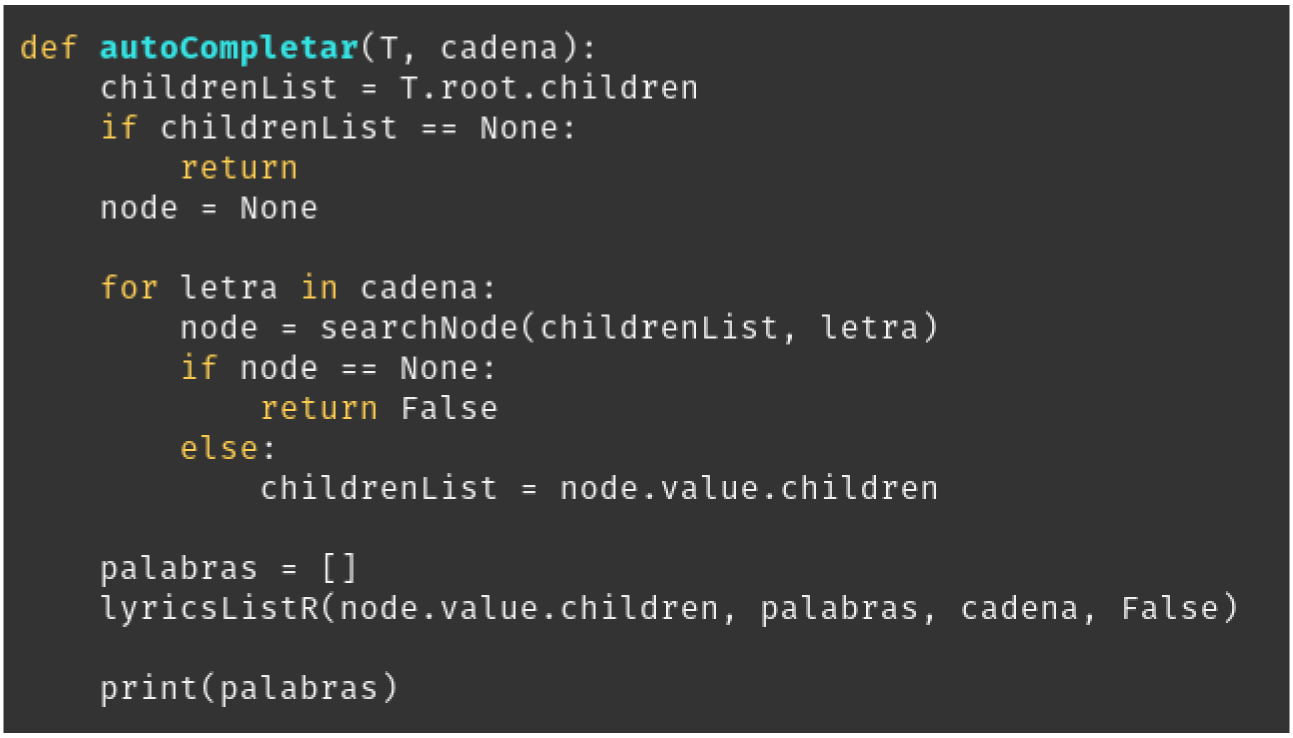
O(n^2)

Ejercicio 6



```python
def invertedString(T):
    strings = []
    lyricsListR(T.root.children, strings, "", False)

    for element in strings:
        elementRevert = ""
        for lyric in reversed(element):
            elementRevert += lyric
        for element2 in strings:
            if elementRevert == element2:
                return True

    return False
```

Ejercicio7



```python
def autoCompletar(T, cadena):
    childrenList = T.root.children
    if childrenList == None:
        return
    node = None

    for letra in cadena:
        node = searchNode(childrenList, letra)
        if node == None:
            return False
        else:
            childrenList = node.value.children

    palabras = []
    lyricsListR(node.value.children, palabras, cadena, False)

    print(palabras)
```