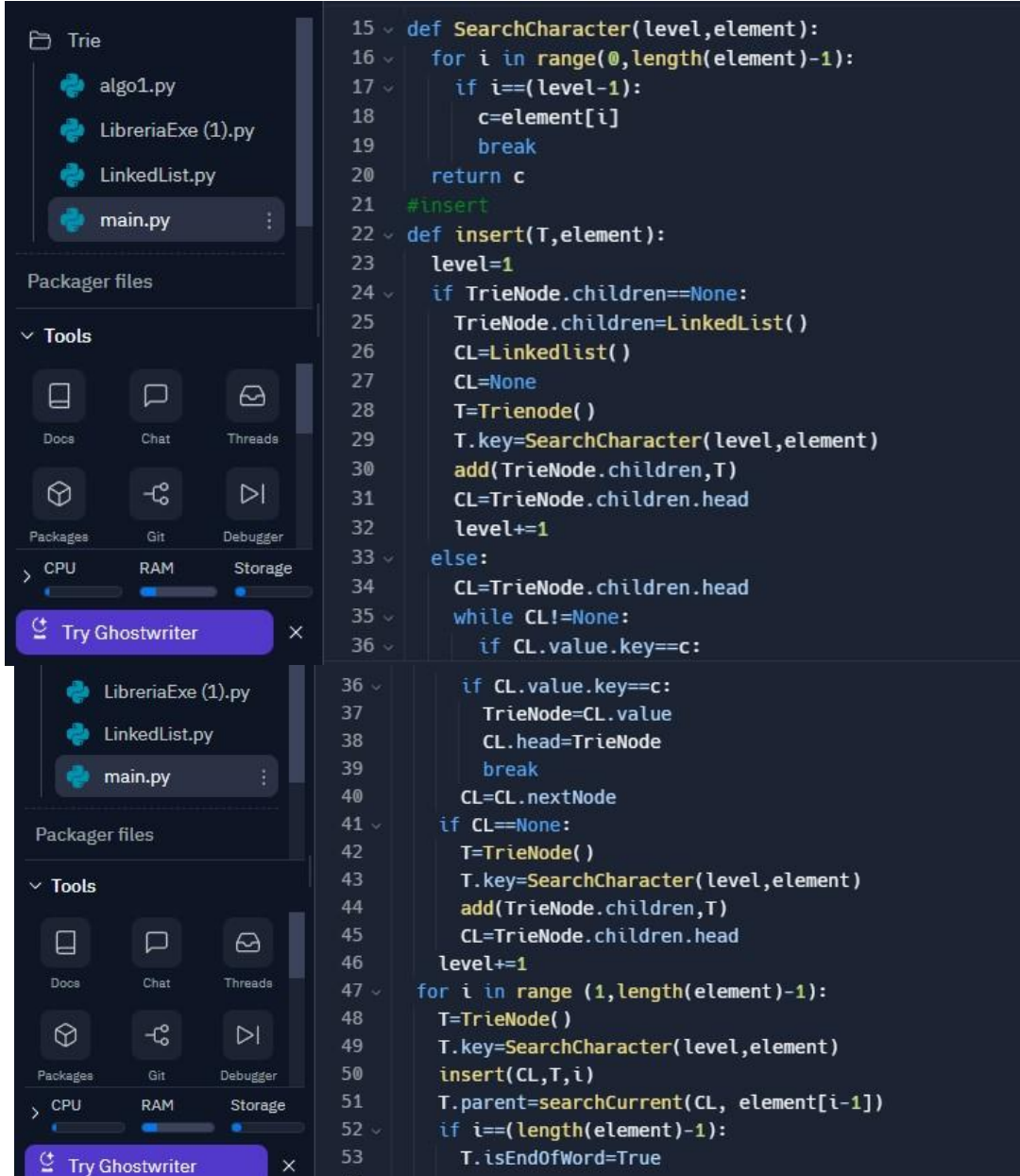


Algoritmos II

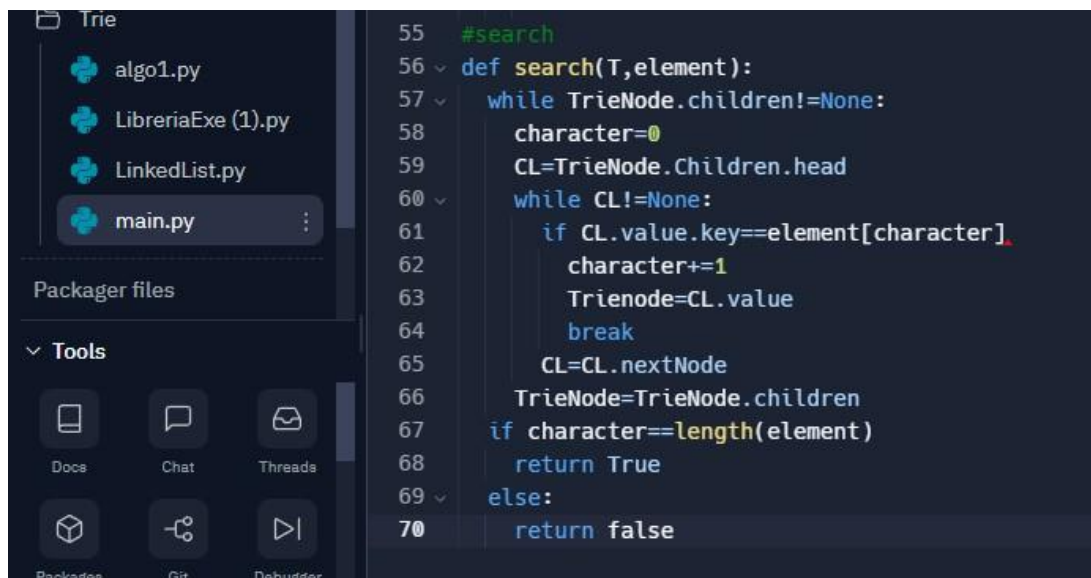
Trabajo Practico arboles N-arios-Trie

1) a) Insert:



```
15 def SearchCharacter(level,element):
16     for i in range(0,length(element)-1):
17         if i==(level-1):
18             c=element[i]
19             break
20     return c
21 #insert
22 def insert(T,element):
23     level=1
24     if TrieNode.children==None:
25         TrieNode.children=LinkedList()
26         CL=LinkedList()
27         CL=None
28         T=TrieNode()
29         T.key=SearchCharacter(level,element)
30         add(TrieNode.children,T)
31         CL=TrieNode.children.head
32         level+=1
33     else:
34         CL=TrieNode.children.head
35         while CL!=None:
36             if CL.value.key==c:
37                 if CL.value.key==c:
38                     TrieNode=CL.value
39                     CL.head=TrieNode
40                     break
41                 CL=CL.nextNode
42             if CL==None:
43                 T=TrieNode()
44                 T.key=SearchCharacter(level,element)
45                 add(TrieNode.children,T)
46                 CL=TrieNode.children.head
47                 level+=1
48         for i in range (1,length(element)-1):
49             T=TrieNode()
50             T.key=SearchCharacter(level,element)
51             insert(CL,T,i)
52             T.parent=searchCurrent(CL, element[i-1])
53             if i==(length(element)-1):
54                 T.isEndOfWord=True
```

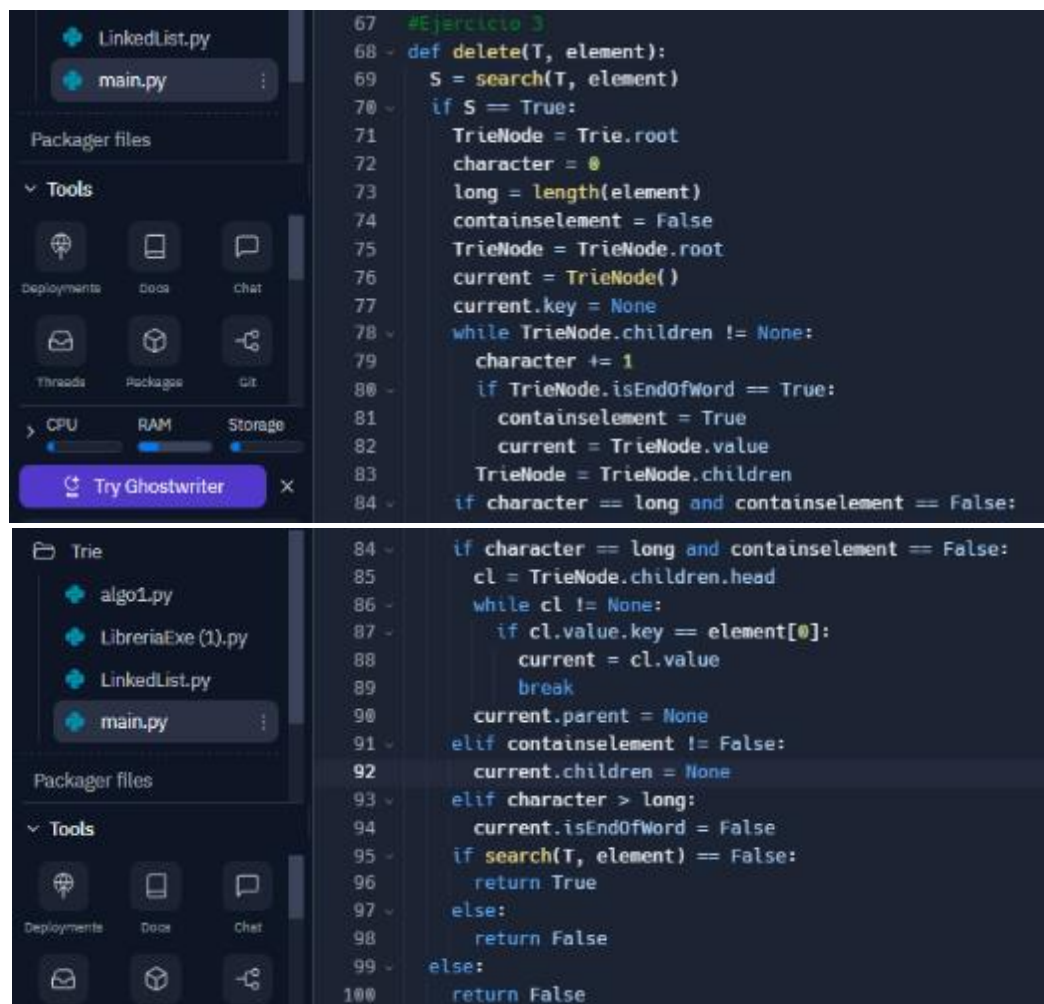
B) Search:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'Trie' containing files 'algo1.py', 'LibreriaExe (1).py', 'LinkedList.py', and 'main.py'. The code editor shows the following Python code:

```
55 #search
56 def search(T,element):
57     while TrieNode.children!=None:
58         character=0
59         CL=TrieNode.Children.head
60         while CL!=None:
61             if CL.value.key==element[character]:
62                 character+=1
63                 Trienode=CL.value
64                 break
65             CL=CL.nextNode
66         TrieNode=TrieNode.children
67     if character==length(element):
68         return True
69     else:
70         return false
```

3)



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'Trie' containing files 'algo1.py', 'LibreriaExe (1).py', 'LinkedList.py', and 'main.py'. The code editor shows the following Python code:

```
67 #Ejercicio 3
68 def delete(T, element):
69     S = search(T, element)
70     if S == True:
71         TrieNode = Trie.root
72         character = 0
73         long = length(element)
74         containselement = False
75         TrieNode = TrieNode.root
76         current = TrieNode()
77         current.key = None
78         while TrieNode.children != None:
79             character += 1
80             if TrieNode.isEndOfWord == True:
81                 containselement = True
82                 current = TrieNode.value
83                 TrieNode = TrieNode.children
84             if character == long and containselement == False:
85                 if character == long and containselement == False:
86                     cl = TrieNode.children.head
87                     while cl != None:
88                         if cl.value.key == element[0]:
89                             current = cl.value
90                             break
91                     current.parent = None
92                 elif containselement != False:
93                     current.children = None
94                 elif character > long:
95                     current.isEndOfWord = False
96                     if search(T, element) == False:
97                         return True
98                     else:
99                         return False
100         else:
101             return False
```

4)

```

101  '''Parte 2'''
102  #Ejercicio 4
103  def Printwords(Trie,P,N):
104      tp=searchcharacter(Trie, P)
105      long=N
106      if tp!=None:
107          c=tp.children.head
108          while c!=None:
109              L=LinkedList()
110              L=PrintwordsR(Trie,tp,c,long)
111              print_list(L)
112              c=c.nextnode
113  def PrintwordsR(Trie,tp,c,long):
114      T=PrefT(Trie,tp)
115      while T!=None:
116          List=PrefL(Trie,tp,L)
117          long=long-PrefN(Trie,tp)
118          current=T.children.head
119          T=PrefT(Trie,T)
120          if long==1 and T.isEndOfWord==True:
121              return(L)
122              break
123          elif long==1 and T.isEndOfWord==False:
124              break
125          PrintwordsR(Trie,T,current.nextNode,long)
126
127

```

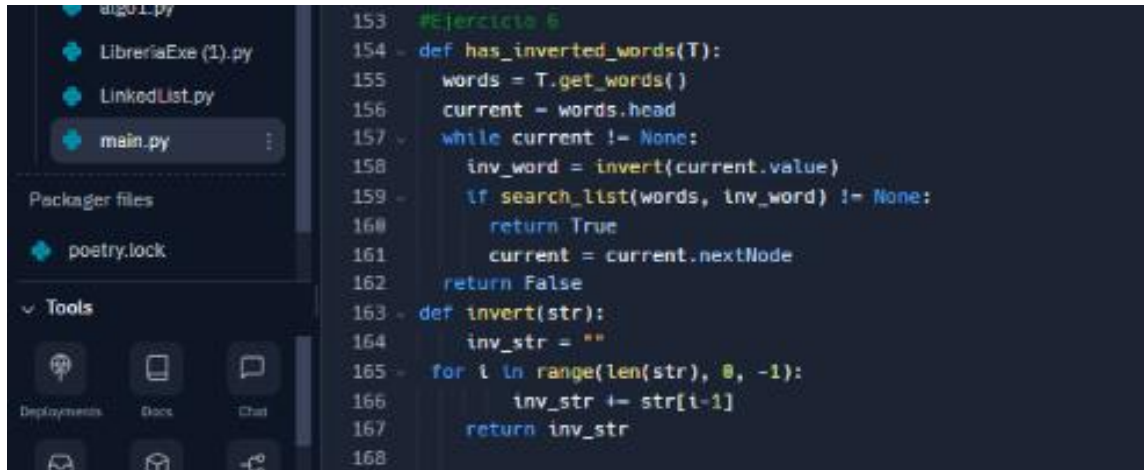
5)

```

120
127  #Ejercicio 5
128  #Recorre el Trie y almacena las palabras
129  def traverse(current,prefijo,elements,pos):
130      if current.isEndOfWord==True:
131          insert(elements,prefijo,pos)
132      for i in range (len(current.children)):
133          traverse(current.children[i],prefijo+ current.children[i].key,palabras,pos+1)
134
135  def save_elements(T):
136      elements = []
137      pos=0
138      traverse(T.root, " ", elements)
139      return elements
140  #verifica que los dos trie sean del mismo documento
141  def are_from_the_same_document(T1, T2):
142      words1 = T1.get_words()
143      words2 = T2.get_words()
144      return is_sublist(words1, words2)
145
146  def is_sublist(list1, list2):
147      current1 = list1.head
148      while current1 != None:
149          if search(list2, current1.value) == None:
150              return False
151          current1 = current1.nextNode
152      return True

```

6)

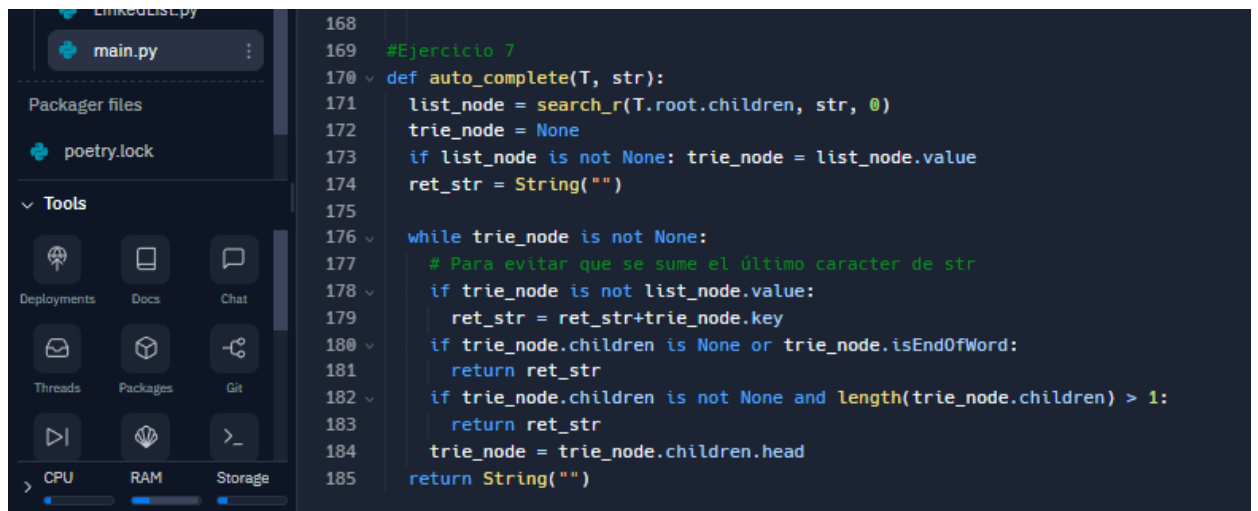


```

153 #Ejercicio 6
154 def has_inverted_words(T):
155     words = T.get_words()
156     current = words.head
157     while current != None:
158         inv_word = invert(current.value)
159         if search_list(words, inv_word) != None:
160             return True
161         current = current.nextNode
162     return False
163 def invert(str):
164     inv_str = ""
165     for i in range(len(str), 0, -1):
166         inv_str += str[i-1]
167     return inv_str
168

```

7)



```

168
169 #Ejercicio 7
170 def auto_complete(T, str):
171     list_node = search_r(T.root.children, str, 0)
172     trie_node = None
173     if list_node is not None: trie_node = list_node.value
174     ret_str = String("")
175
176     while trie_node is not None:
177         # Para evitar que se sume el último caracter de str
178         if trie_node is not list_node.value:
179             ret_str = ret_str+trie_node.key
180         if trie_node.children is None or trie_node.isEndOfWord:
181             return ret_str
182         if trie_node.children is not None and length(trie_node.children) > 1:
183             return ret_str
184         trie_node = trie_node.children.head
185     return String("")

```