```
1     #CMPS 455 Assignment No. 8
2     #Authors: Koppany Horvath
3     #Language: Python 3.6
4     #Task: Given the following CFG and the LR Parsing table. Write a program
      to trace input strings (1) (i+i)*i$ (2) (i*)$. A sample I/O is shown on
      the back
5
6     def matches(string):
7           parseTable = [
8                 {'i':'s5','+':None,'-':None,'*':None,'/':None,'(':'s4',
      ')':None, '$':None,'E':'1', 'T':'2', 'F':'3' }, #0
9                 {'i':None,'+':'s6','-':'s7','*':None,'/':None,'(':None,
      ')':None, '$':True,'E':None,'T':None,'F':None}, #1
10                {'i':None,'+':'r3','-':'r3','*':'s8','/':'s9','(':None,
      ')':'r3', '$':'r3','E':None,'T':None,'F':None}, #2
11                {'i':None,'+':'r6','-':'r6','*':'r6','/':'r6','(':None,
      ')':'r6', '$':'r6','E':None,'T':None,'F':None}, #3
12                {'i':'s5','+':None,'-':None,'*':None,'/':None,'(':'s4',
      ')':None, '$':None,'E':'10','T':'2' ,'F':'3' }, #4
13                {'i':None,'+':'r8','-':'r8','*':'r8','/':'r8','(':None,
      ')':'r8', '$':'r8','E':None,'T':None,'F':None}, #5
14                {'i':'s5','+':None,'-':None,'*':None,'/':None,'(':'s4',
      ')':None, '$':None,'E':None,'T':'11','F':'3' }, #6
15                {'i':'s5','+':None,'-':None,'*':None,'/':None,'(':'s4',
      ')':None, '$':None,'E':None,'T':'12','F':'3' }, #7
16                {'i':'s5','+':None,'-':None,'*':None,'/':None,'(':'s4',
      ')':None, '$':None,'E':None,'T':None,'F':'13'}, #8
17                {'i':'s5','+':None,'-':None,'*':None,'/':None,'(':'s4',
      ')':None, '$':None,'E':None,'T':None,'F':'14'}, #9
18                {'i':None,'+':'s6','-':'s7','*':None,'/':None,'(':None,
      ')':'s15','$':None,'E':None,'T':None,'F':None}, #10
19                {'i':None,'+':'r1','-':'r1','*':'s8','/':'s9','(':None,
      ')':'r1', '$':'r1','E':None,'T':None,'F':None}, #11
20                {'i':None,'+':'r2','-':'r2','*':'s8','/':'s9','(':None,
      ')':'r2', '$':'r2','E':None,'T':None,'F':None}, #12
21                {'i':None,'+':'r4','-':'r4','*':'r4','/':'r4','(':None,
      ')':'r4', '$':'r4','E':None,'T':None,'F':None}, #13
22                {'i':None,'+':'r5','-':'r5','*':'r5','/':'r5','(':None,
      ')':'r5', '$':'r5','E':None,'T':None,'F':None}, #14
23                {'i':None,'+':'r7','-':'r7','*':'r7','/':'r7','(':None,
      ')':'r7', '$':'r7','E':None,'T':None,'F':None}, #15
24          ]
25          cfg = [
26                None,
27                {"nTerm": "E", "popLen": 6}, #(1) E=E+T
28                {"nTerm": "E", "popLen": 6}, #(2) E=E-T
29                {"nTerm": "E", "popLen": 2}, #(3) E=T
30                {"nTerm": "T", "popLen": 6}, #(4) T=T*F
31                {"nTerm": "T", "popLen": 6}, #(5) T=T/F
32                {"nTerm": "T", "popLen": 2}, #(6) T=F
33                {"nTerm": "F", "popLen": 6}, #(7) F=(E)
34                {"nTerm": "F", "popLen": 2}, #(8) F=i
35          ]
36          stack = []
37          curTerm = None
38          curNonTerm = None
39          curIndex = 0
```

```
40          canRead = True
41
42          stack.append(0)
43
44          while True:
45                curIndex = stack.pop()
46                if(canRead):
47                      curTerm = string[0]
48                      string = string[1:]
49                      curNonTerm = curTerm
50                      print("Read:", curTerm, " - ", "Stack:", stack)
51                item = parseTable[curIndex][curNonTerm]
52                if item == None: return False
53                elif item == True: return True
54                elif item[0] == 's':
55                      canRead = True
56                      stack.append(curIndex)
57                      stack.append(curNonTerm)
58                      stack.append(int(item[1:]))
59                elif item[0] == 'r':
60                      canRead = False
61                      stack.append(curIndex)
62                      item = cfg[int(item[1:])]
63                      curNonTerm = item["nTerm"]
64                      for _ in range(item["popLen"]): stack.pop()
65                else:
66                      stack.append(curIndex)
67                      stack.append(curNonTerm)
68                      stack.append(int(item))
69                      curNonTerm = curTerm
70          return False
71
72    for s in ["(i+i)*i$", "(i*)$"]:
73          print("Working on string:", s)
74          if matches(s): print("String matches grammar!")
75          else: print("Error: string does not match grammar!")
76          print()
77
78    """ Output:
79    Working on string: (i+i)*i$
80    Read: (  -  Stack: []
81    Read: i  -  Stack: [0, '(']
82    Read: +  -  Stack: [0, '(', 4, 'i']
83    Read: i  -  Stack: [0, '(', 4, 'E', 10, '+']
84    Read: )  -  Stack: [0, '(', 4, 'E', 10, '+', 6, 'i']
85    Read: *  -  Stack: [0, '(', 4, 'E', 10, ')']
86    Read: i  -  Stack: [0, 'T', 2, '*']
87    Read: $  -  Stack: [0, 'T', 2, '*', 8, 'i']
88    String matches grammar!
89
90    Working on string: (i*)$
91    Read: (  -  Stack: []
92    Read: i  -  Stack: [0, '(']
93    Read: *  -  Stack: [0, '(', 4, 'i']
94    Read: )  -  Stack: [0, '(', 4, 'T', 2, '*']
95    Error: string does not match grammar!
96    """
```