Theory Of Automata

Assignment 3

Group Members:

- Muhammad Talha Bilal (21K-3349)
- Muhammad Hamza (21K-4579)
- Muhammad Salar (21K-4619)

Code Snippet:

```
"""
# ***Theory Of Automata***

# Assignment 3

**Group Members:**

* Muhammad Talha Bilal (21K-3349)
* Muhammad Hamza (21K-4579)
* Muhammad Salar (21K-4619)
"""

class Parser:
    def __init__(self, input_str):
        self.input_str = input_str
        self.pos = 0
        self.out = ""

    def parse(self):
        if self.E():
            if self.pos == len(self.input_str):
                return True
        return False

    def E(self):
        if self.T():
            print('T->E')
            self.out = self.out[:-1]
            self.out += 'E'
            print(self.out+self.input_str[self.pos:len(self.input_str)] + "\n")
            while self.pos < len(self.input_str) and self.input_str[self.pos] == '+':
                self.out += '+'
                self.pos += 1
                if not self.T():
                    return False
                print('E+T->E')
                self.out = self.out[:-1]
                self.out = self.out[:-1]
                self.out = self.out[:-1]
                self.out += 'E'
                print(self.out+self.input_str[self.pos:len(self.input_str)] + "\n")
            return True
        return False
```

```
    def T(self):
        if self.F():
            print('F->T')
            self.out = self.out[:-1]
            self.out += 'T'
            print(self.out+self.input_str[self.pos:len(self.input_str)] + "\n")
            while self.pos < len(self.input_str) and self.input_str[self.pos] == '*':
                self.out += '*'
                self.pos += 1
                if not self.F():
                    return False
                print('T*F->T')
                self.out = self.out[:-1]
                self.out = self.out[:-1]
                self.out = self.out[:-1]
                self.out += 'T'
                print(self.out+self.input_str[self.pos:len(self.input_str)] + "\n")
            return True
        return False


    def F(self):
        if self.pos < len(self.input_str):
            if self.input_str[self.pos] == '(':
                self.out += '('
                self.pos += 1
                if not self.E():
                    return False
                if self.pos < len(self.input_str) and self.input_str[self.pos] == ')':
                    self.out += ')'
                    print('(E)->F')
                    self.out = self.out[:-1]
                    self.out = self.out[:-1]
                    self.out = self.out[:-1]
                    self.out += 'F'
                    print(self.out+self.input_str[self.pos + 1:len(self.input_str)] + "\n")
                    self.pos += 1
                    return True
                else:
                    return False
            elif self.input_str[self.pos] == 'a':
                print('a->F')
                self.pos += 1
                self.out += 'F'
                print(self.out+self.input_str[self.pos:len(self.input_str)] + "\n")
                return True
        return False

ch = "Yes"
while(ch != "No" and ch != "no"):
  #sample inputs:
  #(a+a)*a
  #a*a*
  expression = input("\nEnter a expression to parse: ")
  parser = Parser(expression)
  if parser.parse():
      print("\n(" + expression + ")->E")
  else:
      print('\nIncorrect Structure!')
  ch = input("\nDo you want to continue?\n")
```

Sample Outputs:

```
Enter a expression to parse: a*a*
a->F
F*a*

F->T
T*a*

a->F
T*F*

T*F->T
T*

Incorrect Structure!
```

```
Enter a expression to parse: (a+a)*a
a->F
(F+a)*a

F->T
(T+a)*a

T->E
(E+a)*a

a->F
(E+F)*a

F->T
(E+T)*a

E+T->E
(E)*a

(E)->F
F*a

F->T
T*a

E+T->E
(E)*a

(E)->F
F*a

F->T
T*a

a->F
T*F

T*F->T
T

T->E
E


((a+a)*a)->E

The Expression  (a+a)*a  is a valid expression
```