- **shaista bibi**
- **BSAI_016**

```
#linear grammar
s-->aA
A-->aA|bbB
B-->bC
C-->S
```

1st method for linear grammar implementation

```
class linear_grammar(object):
      #initialize all variable when calling the class DFA
    def __init__(self, non_terminal,terminal ,lenguage, start_non_terminal,final_non_terminal):
        self.non_terminal = non_terminal;
        self.terminal = terminal;
        self.lenguage = lenguage;
        self.start_non_terminal = start_non_terminal;
        self.final_non_terminal = final_non_terminal;
        return;
    def accept_state(self,string):
      non_terminal ='S'
      for i in string:
        if non_terminal=='S':
          if i=='a':
            non_terminal='A'
          else:
            return False
        elif non_terminal=='A':
          if i=='a':
            non_terminal='A'
          elif i=='b':
            non_terminal='E'
          else:
            return False
        elif non_terminal=='E':
          if i=='b':
            non_terminal='B'
          else:
            return False
        elif non_terminal=='B':
           if i=='b':
             non_terminal='C'
           else:
             return False
        elif non_terminal=='C':
           if i=='e':
             non_terminal='S'
             return True
           elif non_terminal == final_non_terminal:
               return True
           else:
             return False
        else:
             return False

linear_gram =linear_grammar(
non_terminal=['S','A','E','B','C'],
terminal=['a','b','e'],
start_non_terminal='S',
final_non_terminal=['S'],
lenguage={'S':'aA','A':'aA','A':'bE','E':'bB','B':'bC','C':'e'}
)
if __name__ == "__main__" :
  string={0:'aabbbe',
          1:'aaaabbbe',
          2:'aaaaebbb'}
  for j in string:
    if(linear_gram.accept_state(string[j])):
        print(string[j],"->","This String  is ACCEPTED.")
    else:
        print(string[j],"->","This String is INVALID.")
```

```
          print(string[j],' ', 'this String is INVALID.')
```

```
aabbbe -> This String  is ACCEPTED.
aaaabbbe -> This String  is ACCEPTED.
aaaaebbb -> This String is INVALID.
```

2nd method for linear grammar implementation

```
class linear_grammar(object):
      #initialize all variable when calling the class DFA
    def __init__(self, non_terminal,terminal ,lenguage, start_non_terminal,final_non_terminal):
        self.non_terminal = non_terminal;
        self.terminal = terminal;
        self.lenguage = lenguage;
        self.start_non_terminal = start_non_terminal;
        self.final_non_terminal = final_non_terminal;
        return;
    def state_S(self,terminal,string):
      for terminal in terminal:
        if (terminal == 'a'):
         state = state_A(self,terminal,string)
         return state
        else:
         return False
    def state_A(self,terminal,string):
      for terminal in terminal:
       if (terminal == 'a'):
         state = state_A(self,terminal,string)
         return state
       elif (terminal == 'b'):
          state = state_empty(self,terminal,string)
          return state
      else:
          return False
    def state_empty(self,terminal,string):
      for terminal in terminal:
          if (terminal == 'b'):
             state = state_B(self,terminal,string)
             return state
          else:
             return False
    def state_B(self,terminal,string):
      for terminal in terminal:
           if (terminal == 'b'):
              state = state_C(self,terminal,string)
              return state
           else:
              return False

    def state_C(self,terminal,string):
      for terminal in terminal:
            if (terminal == 'e'):
              state =state_S(self)
              return True
            else:
               return False
    def accept_string(self,string):
      non_terminal ='S'
      for i in string:
        if non_terminal=='S':
          if i=='a':
            non_terminal='A'
          else:
            return False
        elif non_terminal=='A':
          if i=='a':
            non_terminal='A'
          elif i=='b':
            non_terminal='E'
          else:
            return False
        elif non_terminal=='E':
          if i=='b':
            non_terminal='B'
          else:
```

```
            return False
        elif non_terminal=='B':
           if i=='b':
             non_terminal='C'
           else:
             return False
        elif non_terminal=='C':
           if i=='e':
             non_terminal='S'
             return True
           elif non_terminal == final_non_terminal:
               return True
           else:
             return False
        else:
             return False

linear_gram =linear_grammar(
non_terminal=['S','A','E','B','C'],
terminal=['a','b','e'],
start_non_terminal='S',
final_non_terminal=['S'],
lenguage={'S':'aA','A':'aA','A':'bE','E':'bB','B':'bC','C':'eS'}
)
if __name__ == "__main__" :
  string={0:'aabbbe',
          1:'aaaaebbb'
          }
  for j in string:
    if(linear_gram.accept_string(string[j])):
        print(string[j],"->","This string is ACCEPTED")
    else:
        print(string[j],"->"," Ths string is INVALID")
```

```
 aabbbe -> This string is ACCEPTED
 aaaaebbb ->  Ths string is INVALID
```