```
 1 import string
 2
 3 address = '.\\..\\..\\Grammars\\1\\'
 4
 5
 6 def print_grammar(production, text=None):
 7     print(text)
 8     for key, value in production.items():
 9         print(key, "->", value)
10
11
12 # set list of variables and removed used
13 def set_variables():
14     vars_list = []
15     for s in open(address + "variables.txt", "r").readline():
16         if s in all_variables:
17             all_variables.remove(s)
18             vars_list.append(s)
19     return vars_list
20
21
22 # set grammars from file
23 def set_grammar():
24     file = open(address + "grammar.txt", "r")
25     production_dict = dict()
26     for r in file.readlines():
27         r = r.replace("\n", "").replace(";", "")
28         production_dict[r.split("-")[0]] = r.split(">")[1].split("|")
29     return production_dict
30
31
32 # add grammar to remove unit production
33 def add_grammar(productions, checked, key, vars_list):
34     products = []
35     for p in productions[key]:
36         if len(p) is 1 and p in vars_list:
37             if p not in checked:
38                 checked.append(p)
39                 products.extend(add_grammar(productions, checked, p,
   vars_list))
40         else:
41             products.append(p)
42     return products
43
44
45 def remove_unit_grammar(productions, vars_list):
46     deleted = dict()
47     for key in productions:
```

```
48          checked_list = []
49          deleted[key] = list()
50          for value in productions[key]:
51              if len(value) is 1 and value in vars_list:
52                  checked_list.append(key)
53                  productions[key].extend(add_grammar(productions,
   checked_list, value, vars_list))  # add grammars
54                  deleted[key].append(value)
55                  productions[key] = list(dict.fromkeys(productions[key]))
    # remove duplicates
56      for key, value in deleted.items():
57          for v in value:
58              productions[key].remove(v)  # remove unit parts
59
60
61  # break productions to smaller part
62  def split_productions(productions, all_variables):
63      temp = dict()
64      flag = True
65      for key in productions:
66          for i in range(len(productions[key])):
67              value = productions[key][i]
68              if len(value) > 2:
69                  flag = False
70                  productions[key][i] = value[0:1] + all_variables[0]
71                  x = list()
72                  x.append(value[1:])
73                  temp[all_variables.pop(0)] = x
74      productions.update(temp)
75      return flag
76
77
78  # change terminal with non-terminal
79  def remove_terminal_production(productions, terminals, variables,
   all_variables):
80      terminals_to_variables = dict()
81      for t in terminals.readlines():
82          for temp in t.split(","):
83              terminals_to_variables[all_variables[0]] = list(temp)
84              variables.append(all_variables.remove(all_variables[0]))
85      for key in productions:
86          for i in range(len(productions[key])):
87              value = productions[key][i]
88              if len(value) is 1: continue
89              temp = value
90              for t_k in terminals_to_variables:
91                  temp = temp.replace(terminals_to_variables[t_k][0], t_k)
92              productions[key][i] = temp
```

```
 93     productions.update(terminals_to_variables)
 94
 95
 96 # set values
 97 all_variables = list(string.ascii_uppercase)
 98 terminals = open(address + "terminals.txt", "r")
 99 variables = set_variables()
100 # step 1 set grammar
101 grammars = set_grammar()  # tested
102 print_grammar(grammars, "Grammar is:")
103 # step 2 eliminate unit productions
104 remove_unit_grammar(grammars, variables)  # tested
105 print_grammar(grammars, 'Removed unit production:')
106 # step 3 replacing terminals on RHS with non-terminals
107 remove_terminal_production(grammars, terminals, variables, all_variables
    )  # tested
108 print_grammar(grammars, 'Removed terminals production:')
109 # step 4 split RHS has more than two symbols
110 while True:
111     if split_productions(grammars, all_variables):  # tested
112         break
113 print_grammar(grammars, "CNF is:")
```