example2-output

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java" "-javaagent:D:\Software2\IntelliJ IDEA 2017.3.2\lib\idea_rt
.jar=58952:D:\Software2\IntelliJ IDEA 2017.3.2\bin" -Dfile.encoding=UTF-8 -classpath "C:\Program Files\
Java\jdk1.8.0_181\jre\lib\charsets.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\deploy.jar;C:\Program
Files\Java\jdk1.8.0_181\jre\lib\ext\access-bridge-64.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\ext\
cldrdata.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\ext\dnsns.jar;C:\Program Files\Java\jdk1.8.0_181\
jre\lib\ext\jaccess.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\ext\jfxrt.jar;C:\Program Files\Java\
jdk1.8.0_181\jre\lib\ext\localedata.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\ext\nashorn.jar;C:\
Program Files\Java\jdk1.8.0_181\jre\lib\ext\sunec.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\ext\
sunjce_provider.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\ext\sunmscapi.jar;C:\Program Files\Java\
jdk1.8.0_181\jre\lib\ext\sunpkcs11.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\ext\zipfs.jar;C:\
Program Files\Java\jdk1.8.0_181\jre\lib\javaws.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\jce.jar;C:\
Program Files\Java\jdk1.8.0_181\jre\lib\jfr.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\jfxswt.jar;C:\
Program Files\Java\jdk1.8.0_181\jre\lib\jsse.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\management-
agent.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\plugin.jar;C:\Program Files\Java\jdk1.8.0_181\jre\
lib\resources.jar;C:\Program Files\Java\jdk1.8.0_181\jre\lib\rt.jar;D:\GitOpenProject\Assignment\
compiler2018\compiler\out\production\LR1Grammar" Main

1.INPUT THE START NON_TERMINAL:
>S

2.INPUT THE GRAMMAR:(copy Ɛ from there)
>S->[B
>A->i|[B
>B->]|C
>C->A]|A,C
>end
------------------------all productions
A:
    [[B, i]
B:
    [C, ]]
S:
    [[B]
C:
    [A,C, A]]
----------------------------------first
A:
    [i, []
B:
    [i, [, ]]
S:
    [[]
C:
    [i, []
-----------------------------------DFA
Node 0:
    □-> S    [$]
        ^
    S-> [B    [$]
        ^
Node 1:
    □->S     [$]
         ^
Node 2:
    S->[ B    [$]
         ^
    B-> C    [$]
        ^
    B-> ]    [$]
        ^
    C-> A,C    [$]
        ^
    C-> A]    [$]
        ^
    A-> [B    [,, ]]
        ^
    A-> i    [,, ]]
        ^
Node 3:
    C->A ,C    [$]
         ^
    C->A ]    [$]
         ^
Node 4:
    S->[B     [$]
          ^
Node 5:
    B->C     [$]
         ^
Node 6:
    A->i     [,, ]]
         ^
Node 7:
    A->[ B    [,, ]]
         ^
    B-> C    [,, ]]
        ^
```

example2-output

```
    B-> ]    [,, ]]
        ^
    C-> A,C    [,, ]]
        ^
    C-> A]    [,, ]]
        ^
    A-> [B    [,, ]]
        ^
    A-> i    [,, ]]
        ^
Node 8:
    B->]     [$]
         ^
Node 9:
    C->A, C    [$]
          ^
    C-> A,C    [$]
        ^
    C-> A]    [$]
        ^
    A-> [B    [,, ]]
        ^
    A-> i    [,, ]]
        ^
Node 10:
    C->A]     [$]
          ^
Node 11:
    C->A ,C    [,, ]]
         ^
    C->A ]    [,, ]]
         ^
Node 12:
    A->[B     [,, ]]
          ^
Node 13:
    B->C     [,, ]]
         ^
Node 14:
    B->]     [,, ]]
         ^
Node 15:
    C->A,C     [$]
           ^
Node 16:
    C->A, C    [,, ]]
          ^
    C-> A,C    [,, ]]
        ^
    C-> A]    [,, ]]
        ^
    A-> [B    [,, ]]
        ^
    A-> i    [,, ]]
        ^
Node 17:
    C->A]     [,, ]]
          ^
Node 18:
    C->A,C     [,, ]]
           ^
--------------------------simplified DFA
 node 0:
    via S ==> 1
    via [ ==> 2
 node 1:
    Reduce □->S on [$]
 node 2:
    via A ==> 3
    via B ==> 4
    via C ==> 5
    via i ==> 6
    via [ ==> 7
    via ] ==> 8
 node 3:
    via , ==> 9
    via ] ==> 10
 node 4:
    Reduce S->[B on [$]
 node 5:
    Reduce B->C on [$]
 node 6:
    Reduce A->i on [,, ]]
 node 7:
    via A ==> 11
    via B ==> 12
    via C ==> 13
```

example2-output

```
    via i ==> 6
    via [ ==> 7
    via ] ==> 14
 node 8:
    Reduce B->] on [$]
 node 9:
    via A ==> 3
    via C ==> 15
    via i ==> 6
    via [ ==> 7
 node 10:
    Reduce C->A] on [$]
 node 11:
    via , ==> 16
    via ] ==> 17
 node 12:
    Reduce A->[B on [,, ]]
 node 13:
    Reduce B->C on [,, ]]
 node 14:
    Reduce B->] on [,, ]]
 node 15:
    Reduce C->A,C on [$]
 node 16:
    via A ==> 11
    via C ==> 18
    via i ==> 6
    via [ ==> 7
 node 17:
    Reduce C->A] on [,, ]]
 node 18:
    Reduce C->A,C on [,, ]]
-----------------------------Parse Table
0 {S=go 1, [=shift 2}
1 {$=reduce □->S}
2 {A=go 3, B=go 4, C=go 5, i=shift 6, [=shift 7, ]=shift 8}
3 {,=shift 9, ]=shift 10}
4 {$=reduce S->[B}
5 {$=reduce B->C}
6 {,=reduce A->i, ]=reduce A->i}
7 {A=go 11, B=go 12, C=go 13, i=shift 6, [=shift 7, ]=shift 14}
8 {$=reduce B->]}
9 {A=go 3, C=go 15, i=shift 6, [=shift 7}
10 {$=reduce C->A]}
11 {,=shift 16, ]=shift 17}
12 {,=reduce A->[B, ]=reduce A->[B}
13 {,=reduce B->C, ]=reduce B->C}
14 {,=reduce B->], ]=reduce B->]}
15 {$=reduce C->A,C}
16 {A=go 11, C=go 18, i=shift 6, [=shift 7}
17 {,=reduce C->A], ]=reduce C->A]}
18 {,=reduce C->A,C, ]=reduce C->A,C}

3.INPUT THE SENTENCE:
>[i,i]
STATE_STACK         NOTATION_STACK      INPUT               ACTION
[0]                 [$]                 [i,i]$              shift 2
[0, 2]              [$, []              i,i]$               shift 6
[0, 2, 6]           [$, [, i]           ,i]$                reduce A->i
[0, 2]              [$, [, A]           ,i]$                go 3
[0, 2, 3]           [$, [, A]           ,i]$                shift 9
[0, 2, 3, 9]        [$, [, A, ,]        i]$                 shift 6
[0, 2, 3, 9, 6]     [$, [, A, ,, i]     ]$                  reduce A->i
[0, 2, 3, 9]        [$, [, A, ,, A]     ]$                  go 3
[0, 2, 3, 9, 3]     [$, [, A, ,, A]     ]$                  shift 10
[0, 2, 3, 9, 3, 10] [$, [, A, ,, A, ]]  $                   reduce C->A]
[0, 2, 3, 9]        [$, [, A, ,, C]     $                   go 15
[0, 2, 3, 9, 15]    [$, [, A, ,, C]     $                   reduce C->A,C
[0, 2]              [$, [, C]           $                   go 5
[0, 2, 5]           [$, [, C]           $                   reduce B->C
[0, 2]              [$, [, B]           $                   go 4
[0, 2, 4]           [$, [, B]           $                   reduce S->[B
[0]                 [$, S]              $                   go 1
[0, 1]              [$, S]              $                   reduce □->S
[0]                 [$, □]              $                   Accepted

Process finished with exit code 0
```