```
Opening files...
Parsing specification from standard input...
Checking specification...
Building parse tables...
  Computing non-terminal nullability...
  Computing first sets...
  Building state machine...
  Filling in tables...
  Checking for non-reduced productions...
Writing parser...

===== Terminals =====

[0]EOF [1]error [2]a [3]b [4]c [5]d

===== Non terminals =====

[0]$START [1]S [2]C [3]D

===== Productions =====

[0] S ::= C D C
[1] $START ::= S EOF
[2] C ::= c C
[3] C ::= d
[4] D ::= a D b
[5] D ::= a b
[6] D ::= error

===== Viable Prefix Recognizer =====

START lalr_state [0]: {

  [C ::= (*) c C , {error a }]
  [$START ::= (*) S EOF , {EOF }]
  [C ::= (*) d , {error a }]
  [S ::= (*) C D C , {EOF }]
}

transition on c to state [4]
transition on d to state [3]
transition on C to state [2]
transition on S to state [1]

-------------------

lalr_state [1]: {

  [$START ::= S (*) EOF , {EOF }]
}

transition on EOF to state [13]
```

```
------------------

lalr_state [2]: {

  [D ::= (*) a b , {c d }]
  [D ::= (*) a D b , {c d }]
  [D ::= (*) error , {c d }]
  [S ::= C (*) D C , {EOF }]
}

transition on D to state [8]
transition on error to state [7]
transition on a to state [6]

------------------

lalr_state [3]: {

  [C ::= d (*) , {EOF error a }]
}

------------------

lalr_state [4]: {

  [C ::= c (*) C , {EOF error a }]
  [C ::= (*) c C , {EOF error a }]
  [C ::= (*) d , {EOF error a }]
}

transition on c to state [4]
transition on C to state [5]
transition on d to state [3]

------------------

lalr_state [5]: {

  [C ::= c C (*) , {EOF error a }]
}

------------------

lalr_state [6]: {

  [D ::= a (*) b , {b c d }]
  [D ::= (*) a b , {b }]
  [D ::= a (*) D b , {b c d }]
  [D ::= (*) a D b , {b }]
  [D ::= (*) error , {b }]
}
```

```
transition on D to state [11]
transition on error to state [7]
transition on a to state [6]
transition on b to state [10]

------------------

lalr_state [7]: {

  [D ::= error (*) , {b c d }]
}

------------------

lalr_state [8]: {

  [C ::= (*) c C , {EOF }]
  [S ::= C D (*) C , {EOF }]
  [C ::= (*) d , {EOF }]
}

transition on c to state [4]
transition on C to state [9]
transition on d to state [3]

------------------

lalr_state [9]: {

  [S ::= C D C (*) , {EOF }]
}

------------------

lalr_state [10]: {

  [D ::= a b (*) , {b c d }]
}

------------------

lalr_state [11]: {

  [D ::= a D (*) b , {b c d }]
}

transition on b to state [12]
```

```
------------------

lalr_state [12]: {

  [D ::= a D b (*) , {b c d }]
}

------------------

lalr_state [13]: {

  [$START ::= S EOF (*) , {EOF }]
}

------------------

-------- ACTION_TABLE --------

From state #0
 [term 4:SHIFT(to state 4)] [term 5:SHIFT(to state 3)]

From state #1
 [term 0:SHIFT(to state 13)]

From state #2
 [term 1:SHIFT(to state 7)] [term 2:SHIFT(to state 6)]

From state #3
 [term 0:REDUCE(with prod 3)] [term 1:REDUCE(with prod 3)]
 [term 2:REDUCE(with prod 3)]

From state #4
 [term 4:SHIFT(to state 4)] [term 5:SHIFT(to state 3)]

From state #5
 [term 0:REDUCE(with prod 2)] [term 1:REDUCE(with prod 2)]
 [term 2:REDUCE(with prod 2)]

From state #6
 [term 1:SHIFT(to state 7)] [term 2:SHIFT(to state 6)]
 [term 3:SHIFT(to state 10)]

From state #7
 [term 3:REDUCE(with prod 6)] [term 4:REDUCE(with prod 6)]
 [term 5:REDUCE(with prod 6)]

From state #8
 [term 4:SHIFT(to state 4)] [term 5:SHIFT(to state 3)]

From state #9
 [term 0:REDUCE(with prod 0)]
```

From state #10
 [term 3:REDUCE(with prod 5)] [term 4:REDUCE(with prod 5)]
 [term 5:REDUCE(with prod 5)]

From state #11
 [term 3:SHIFT(to state 12)]

From state #12
 [term 3:REDUCE(with prod 4)] [term 4:REDUCE(with prod 4)]
 [term 5:REDUCE(with prod 4)]

From state #13
 [term 0:REDUCE(with prod 1)]

-------- REDUCE_TABLE --------

From state #0
 [non term 1->state 1] [non term 2->state 2]

From state #1

From state #2
 [non term 3->state 8]

From state #3

From state #4
 [non term 2->state 5]

From state #5

From state #6
 [non term 3->state 11]

From state #7

From state #8
 [non term 2->state 9]

From state #9

From state #10

From state #11

From state #12

From state #13

```
Closing files...

------- CUP v0.10k Parser Generation Summary -------

  0 errors and 0 warnings
  6 terminals, 4 non-terminals, and 7 productions declared,
  producing 14 unique parse states.
  0 terminals declared but not used
  0 non-terminals declared but not used.
  0 productions never reduced.
  0 conflicts detected (0 expected).

  Code written to "parser.java", and "sym.java".
```

Cadena a analizar: **cdaabbd**

```
# Initializing parser

# Current Symbol is #4
# Shift under term #4 to state #4
# Current token is #5
# Shift under term #5 to state #3
# Current token is #2
# Reduce with prod #3 [NT=2, SZ=1]
# Reduce rule: top state 4, lhs sym 2 -> state 5
# Goto state #5
# Reduce with prod #2 [NT=2, SZ=2]
# Reduce rule: top state 0, lhs sym 2 -> state 2
# Goto state #2
# Shift under term #2 to state #6
# Current token is #2
# Shift under term #2 to state #6
# Current token is #3
# Shift under term #3 to state #10
# Current token is #3
# Reduce with prod #5 [NT=3, SZ=2]
# Reduce rule: top state 6, lhs sym 3 -> state 11
# Goto state #11
# Shift under term #3 to state #12
# Current token is #5
# Reduce with prod #4 [NT=3, SZ=3]
# Reduce rule: top state 2, lhs sym 3 -> state 8
# Goto state #8
# Shift under term #5 to state #3
# Current token is #0
# Reduce with prod #3 [NT=2, SZ=1]
# Reduce rule: top state 8, lhs sym 2 -> state 9
# Goto state #9
# Reduce with prod #0 [NT=1, SZ=3]
# Reduce rule: top state 0, lhs sym 1 -> state 1
# Goto state #1
# Shift under term #0 to state #13
# Current token is #0
# Reduce with prod #1 [NT=0, SZ=2]
# Reduce rule: top state 0, lhs sym 0 -> state -1
# Goto state #-1

Fin del Análisis
```

Cadena a analizar: **cdabbd**

```
# Initializing parser

# Current Symbol is #4
# Shift under term #4 to state #4
# Current token is #5
# Shift under term #5 to state #3
# Current token is #2
# Reduce with prod #3 [NT=2, SZ=1]
# Reduce rule: top state 4, lhs sym 2 -> state 5
# Goto state #5
# Reduce with prod #2 [NT=2, SZ=2]
# Reduce rule: top state 0, lhs sym 2 -> state 2
# Goto state #2
# Shift under term #2 to state #6
# Current token is #3
# Shift under term #3 to state #10
# Current token is #3
# Reduce with prod #5 [NT=3, SZ=2]
# Reduce rule: top state 2, lhs sym 3 -> state 8
# Goto state #8

Syntax error at character 1 of input

# Attempting error recovery
# Finding recovery state on stack
# Pop stack by one, state was # 8
# Recover state found (#2)
# Shifting on error to state #7
# Trying to parse ahead
# Parse-ahead reduces: handle size = 1 lhs = #3 from state #2
# Goto state #8
# Consuming Symbol #3
# Trying to parse ahead

# Parse-ahead reduces: handle size = 1 lhs = #3 from state #2
# Goto state #8
# Parse-ahead shifts Symbol #5 into state #3
# Parse-ahead reduces: handle size = 1 lhs = #2 from state #8
# Goto state #9
# Parse-ahead reduces: handle size = 3 lhs = #1 from state #0
# Goto state #1
# Parse-ahead shifts Symbol #0 into state #13
# Parse-ahead accepts
# Parse-ahead ok, going back to normal parse

# Reparsing saved input with actions
# Current Symbol is #5
# Current state is #7
```

Error INCORRECTO !!
Reduce with prod #6 [NT=3, SZ=1]
Goto state #8
Shift under term #5 to state #3
Current Symbol is #0
Reduce with prod #3 [NT=2, SZ=1]
Goto state #9
Reduce with prod #0 [NT=1, SZ=3]
Goto state #1
Shift under term #0 to state #13
Current Symbol is #0
Reduce with prod #1 [NT=0, SZ=2]
Goto state #-1

Fin del Análisis