# PRACTICAL 4

Name : Atharva

Rewatkar

Roll No. : 32

Batch : E2

Subject : Compiler Designer Lab

## AIM:

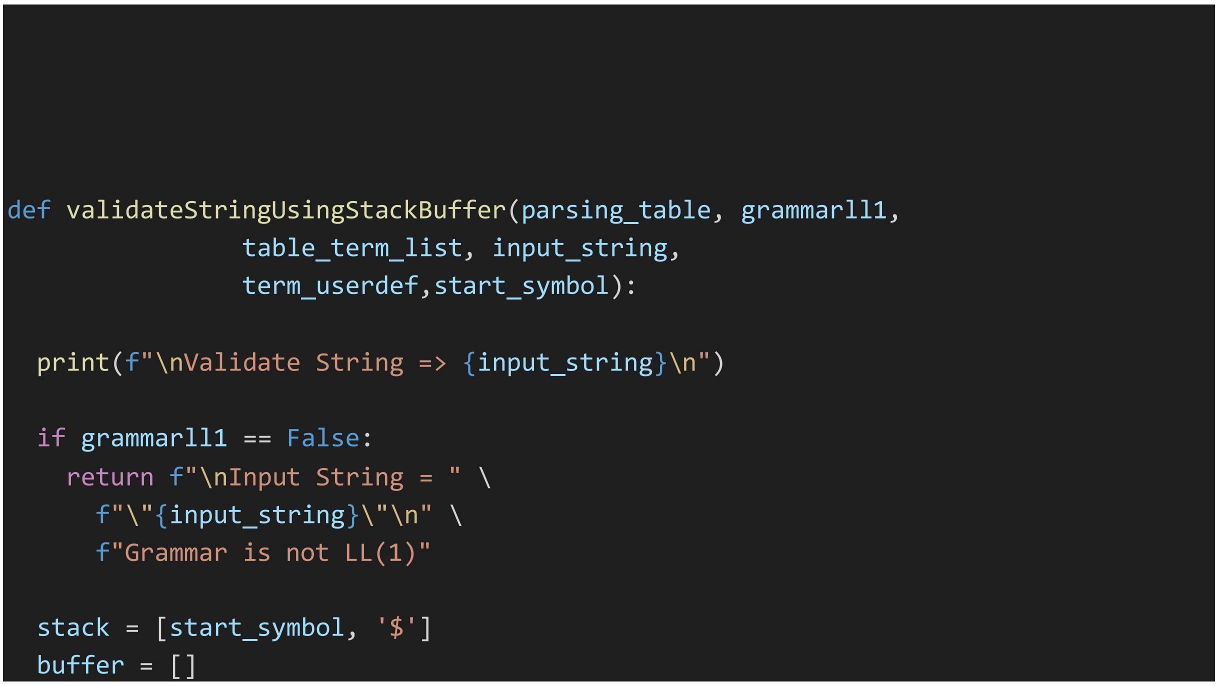
(A) Write a program to validate a natural language sentence. Design a natural language grammar, compute and input the LL (1) table. Validate if the given sentence is valid or not based on the grammar.

Input: NLP grammar and LL (1) parsing table (from file)

Implementation: String parsing rules

Output: Each step-in string parsing and whether the input string is valid or invalid.

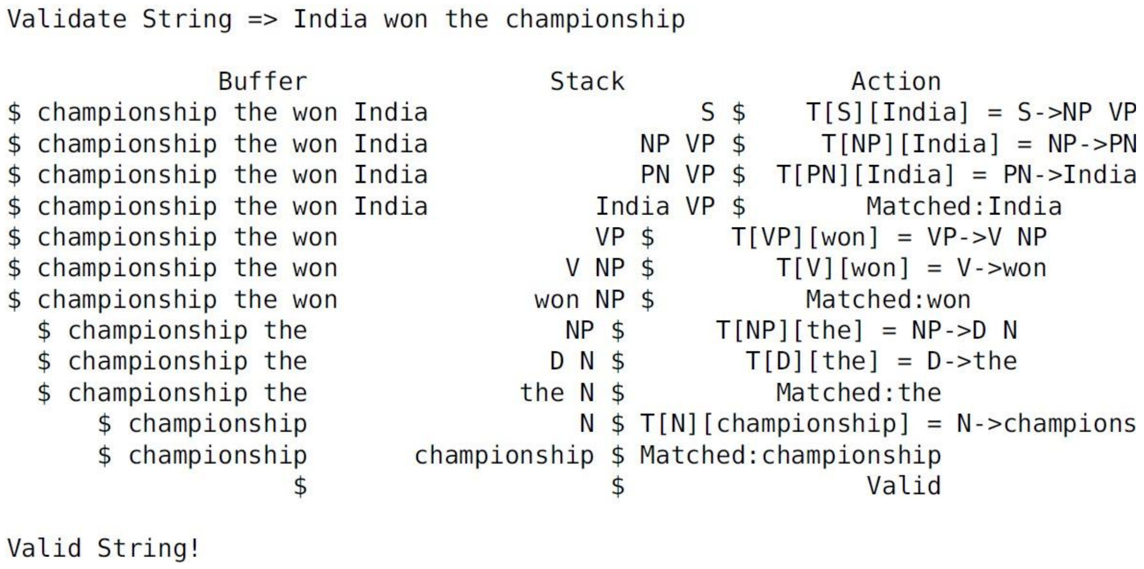
## CODE:



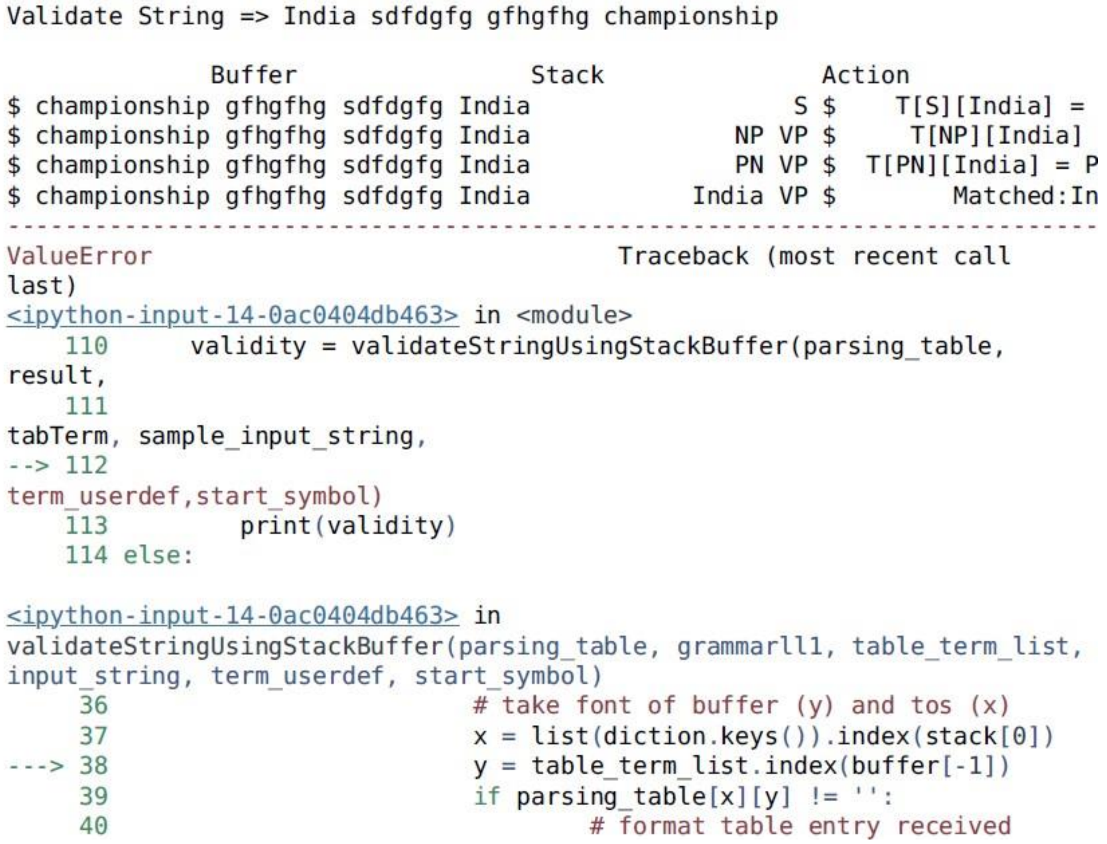
```
def validateStringUsingStackBuffer(parsing_table, grammarll1,
                table_term_list, input_string,
                term_userdef,start_symbol):

  print(f"\nValidate String => {input_string}\n")

  if grammarll1 == False:
    return f"\nInput String = " \
      f"\"{input_string}\"\n" \
      f"Grammar is not LL(1)"

  stack = [start_symbol, '$']
  buffer = []
  while True:
    if stack == ['$'] and buffer == ['$']:
      print("{:>20} {:>20} {:>20}"
        .format(' '.join(buffer),
            ' '.join(stack),
            "Valid"))
      return "\nValid String!"
    elif stack[0] not in term_userdef:

      x = list(diction.keys()).index(stack[0])
      y  =  table_term_list.index(buffer[-1])
      if parsing_table[x][y] != '':

        entry = parsing_table[x][y]
        print("{:>20} {:>20} {:>25}".
          format(' '.join(buffer),
              ' '.join(stack),
              f"T[{stack[0]}][{buffer[-1]}] = {entry}"))
        lhs_rhs = entry.split("->")
        lhs_rhs[1] = lhs_rhs[1].replace('#', '').strip()
        entryrhs = lhs_rhs[1].split()
        stack = entryrhs + stack[1:]
      else:
        return f"\nInvalid String! No rule at " \
          f"Table[{stack[0]}][{buffer[-1]}]."
    else:

      if stack[0] == buffer[-1]:
        print("{:>20} {:>20} {:>20}"
          .format(' '.join(buffer),
              ' '.join(stack),
              f"Matched:{stack[0]}"))
        buffer = buffer[:-1]
        stack = stack[1:]
      else:
        return "\nInvalid String! " \
          "Unmatched terminal symbols"

nonterm_userdef = ['S', 'NP', 'VP', 'N', 'V', 'P', 'PN', 'D']
term_userdef = ["championship", "ball", "toss", "is", "want",
"won", "played", "me", "I", "you", "India",
"Australia","Steve", "John", "the", "a", "an"]
sample_input_string = "India won the championship"
```

```
parsing_table=[['', '', '', '', '', '', '', 'S->NP VP', 'S->NP VP', 'S->NP VP',
'S->NP VP', 'S->NP VP', 'S->NP VP', 'S->NP VP', 'S->NP VP', 'S->NP VP', 'S->NP
VP', ''],
               ['', '', '', '', '', '', '', 'NP->P', 'NP->P', 'NP->P', 'NP->PN',
'NP->PN', 'NP->PN', 'NP->PN', 'NP->D N', 'NP->D N', 'NP->D N', ''],
               ['', '', '', 'VP->V NP', 'VP->V NP', 'VP->V NP', 'VP->V NP', '',
'', '', '', '', '', '', '', '', '', ''],
               ['N->championship', 'N->ball', 'N->toss', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', ''],
               ['', '', '', 'V->is', 'V->want', 'V->won', 'V->played', '', '', '',
'', '', '', '', '', '', '', ''],
               ['', '', '', '', '', '', '', 'P->me', 'P->I', 'P->you', '', '', '',
'', '', '', '', ''],
               ['', '', '', '', '', '', '', '', '', '', 'PN->India', 'PN-
>Australia', 'PN->Steve', 'PN->John', '', '', '', ''],
               ['', '', '', '', '', '', '', '', '', '', '', '', '', '', 'D->the',
'D->a', 'D->an', '']]
result=True
tabTerm=['championship',
'ball',
 'toss',
 'is',
 'want',
 'won',
 'played',
 'me',
 'I',
 'you',
 'India',
 'Australia',
 'Steve',
 'John',
 'the',
 'a',
 'an',
 '$']
start_symbol = 'S'

diction = {}
firsts = {}
follows = {}

if sample_input_string != None:
  validity = validateStringUsingStackBuffer(parsing_table, result,
                      tabTerm, sample_input_string,
                      term_userdef,start_symbol)
  print(validity)
else:
  print("\nNo input String detected")
```

# OUTPUT:

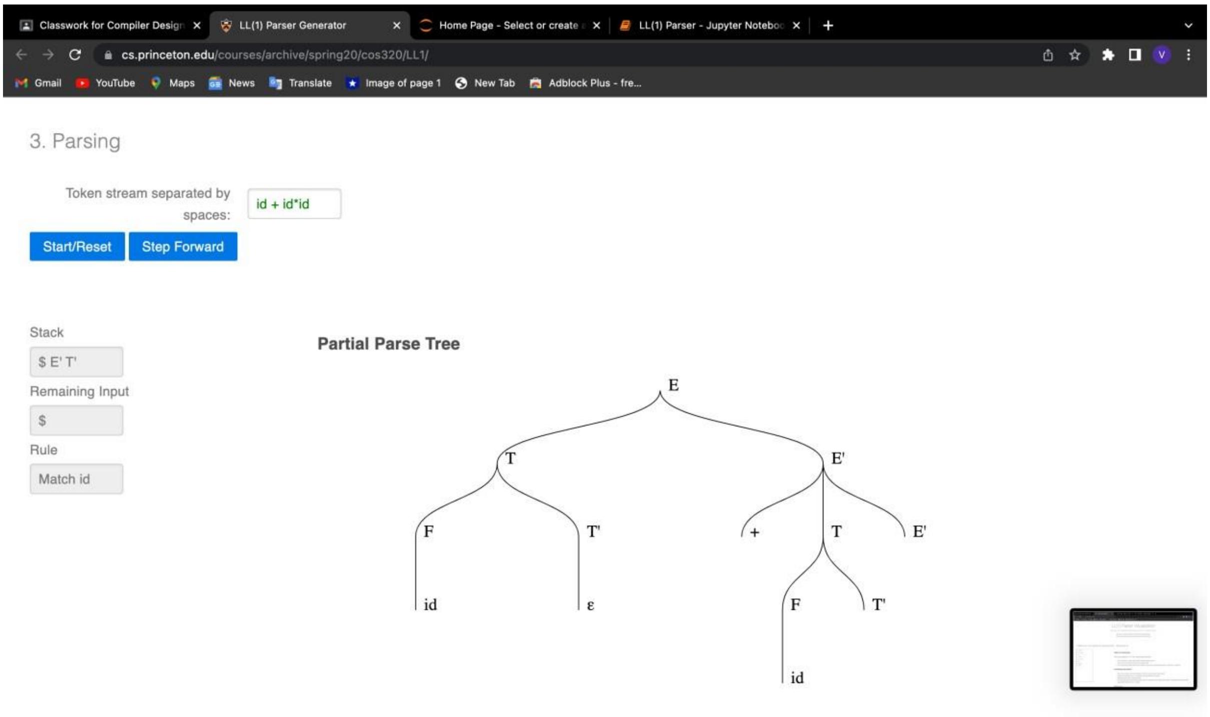


```
Validate String => India won the championship

              Buffer                Stack               Action
$ championship the won India                  S $    T[S][India] = S->NP VP
$ championship the won India              NP VP $     T[NP][India] = NP->PN
$ championship the won India              PN VP $  T[PN][India] = PN->India
$ championship the won India           India VP $         Matched:India
$ championship the won                      VP $     T[VP][won] = VP->V NP
$ championship the won                    V NP $        T[V][won] = V->won
$ championship the won                  won NP $           Matched:won
  $ championship the                        NP $     T[NP][the] = NP->D N
  $ championship the                       D N $       T[D][the] = D->the
  $ championship the                     the N $          Matched:the
      $ championship                         N $ T[N][championship] = N->champions
      $ championship           championship $ Matched:championship
                   $                          $                Valid

Valid String!
```



```
Validate String => India sdfdgfg gfhgfhg championship

              Buffer                Stack               Action
$ championship gfhgfhg sdfdgfg India                  S $    T[S][India] =
$ championship gfhgfhg sdfdgfg India              NP VP $     T[NP][India]
$ championship gfhgfhg sdfdgfg India              PN VP $  T[PN][India] = P
$ championship gfhgfhg sdfdgfg India           India VP $        Matched:In
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call
last)
<ipython-input-14-0ac0404db463> in <module>
    110         validity = validateStringUsingStackBuffer(parsing_table,
result,
    111
tabTerm, sample_input_string,
--> 112
term_userdef,start_symbol)
    113         print(validity)
    114 else:

<ipython-input-14-0ac0404db463> in
validateStringUsingStackBuffer(parsing_table, grammarll1, table_term_list,
input_string, term_userdef, start_symbol)
     36                         # take font of buffer (y) and tos (x)
     37                         x = list(diction.keys()).index(stack[0])
---> 38                         y = table_term_list.index(buffer[-1])
     39                         if parsing_table[x][y] != '':
     40                                 # format table entry received
```

(B) Use Virtual Lab on LL1 parser to validate the string and verify your string validation using simulation.


Classwork for Compiler Design
LL(1) Parser Generator
Home Page - Select or create a
LL(1) Parser - Jupyter Notebook
cs.princeton.edu/courses/archive/spring20/cos320/LL1/
Gmail
YouTube
Maps
News
Translate
Image of page 1
New Tab
Adblock Plus - fre...

3. Parsing

Token stream separated by spaces: id + id*id

Stack: $ E' T'

Remaining Input: $

Rule: Match id

Partial Parse Tree

E
T
E'
F
T'
+
T
E'
id
ε
F
T'
id