Name: Kush Munot
Roll No. 47

Practical No. 4

AIM - 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.

CODE

```
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", "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"]
         ]


def validate(parsing_table, table_term_list, input_string, term_userdef):
    print(f"\nValidate String => {input_string}\n")
    stack = ['S', '$']
    buffer = []
    input_string = input_string.split()
    input_string.reverse()
    buffer = ['$'] + input_string
    print("{:>20} {:>20} {:>40}".format("Buffer", "Stack", "Action"))
    while True:
        # end loop if all symbols matched
        if stack == ['$'] and buffer == ['$']:
            print("{:>20} {:>20} {:>50}".format(
                ' '.join(buffer), ' '.join(stack), "Valid"))
```

```
            return "\nValid String!"
        elif stack[0] not in term_userdef:
            # take front of buffer (y) and tos (x)
            x = list(['S', 'NP', 'VP', 'N', 'V',
                      'P', 'PN', 'D']).index(stack[0])
            y = table_term_list.index(buffer[-1])
            if parsing_table[x][y] != '':
                # format table entry received
                entry = parsing_table[x][y]
                print("{:>20} {:>20} {:>50}".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:
            # stack top is Terminal
            if stack[0] == buffer[-1]:
                print("{:>20} {:>20} {:>50}"
                      .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"]
tabTerm = ["championship", "ball", "toss", "is", "want",
           "won", "played", "me", "I", "you", "India",
           "Australia", "Steve", "John", "the", "a", "an", "$"]
sample_input_string = "Australia won the toss"
validity = validate(table, tabTerm, sample_input_string, term_userdef)
print(validity)
```

Execution Snapshots

```
Kush@Kushs-PC MINGW64 /d/Engg. Third Year/AI-Lab (main)
$ C:/Users/Kush/anaconda3/python.exe "d:/Engg. Third Year/AI-Lab/demo1.py"

Validate String => Australia won the toss

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

Valid String!
(base)
```

```
Kush@Kushs-PC MINGW64 /d/Engg. Third Year/AI-Lab (main)
$ C:/Users/Kush/anaconda3/python.exe "d:/Engg. Third Year/AI-Lab/demo1.py"

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->championship
      $ championship       championship $                          Matched:championship
                   $                  $                                        Valid

Valid String!
(base)
```