Compiler Mini Project

<u>Aim</u>: Write a code to generate a predictive parsing table for a given set of production rules.

What is predictive prasing?

- A predictive parser is a recursive descent parser with no backtracking or backup.
- It is a top-down parser that does not require backtracking.
- At each step, the choice of the rule to be expanded is made upon
- the next terminal symbol.

CODE:

```
• • •
            self.non_terminals = list("EGTUF")
            self.terminals = list("+*()a")
            self.production_rules = ["E→TG", "G→+TG", "G→⊕", "T→FU", "U→+FU", "U→⊕", "F→(E)", "F>a"]
self.first = {"E":["(", "a"], "G":["+", "@"], "T":["(", "a"], "U":["*", "@"], "F":["(", "a"]}
self.follow = {"E":[")", "$"], "G":[")", "$"], "T":[")", "$", "+"], "U":[")", "$", "+"], "F":
    [")", "$", "+", "*"]}
      def generate_parsing_table(self):
            parsing_table = dict()
for non_terminal in self.non_terminals:
    parsing_table[non_terminal] = [None for i in range(len(self.terminals) + 1)]
            for production_rule in self.production_rules:
    non_terminal_at_left, remainder = production_rule.split("→") if "→" in production_rule
    else production_rule.split("-")
                  if not (remainder[0].isupper() or remainder[0] = "@"):
    parsing_table[non_terminal_at_left][self.terminals.index(remainder[0])] =
                         update_locations = self.first[non_terminal_at_left]
                         if "@" in update_locations:
    update_locations.remove("@")
                              update_locations += self.follow[non_terminal_at_left]
                         for update_location in update_locations:
                                    position = self.terminals.index(update_location)
                                     position = len(self.terminals)
                               if parsing_table[non_terminal_at_left][position] is not None:
                               parsing_table[non_terminal_at_left][position] = production_rule
            return parsing_table
            for terminal in self.terminals:
    print(terminal, end = "\t")
                  print(entry, end = "\t\t")
for cell in parsing_table[entry]:
    print(cell, end = "\t")
      predictive_parser = PredictiveParser()
arsing_table = predictive_parser.generate_parsing_table()
      predictive_parser.print_parsing_table(parsing_table)
```

Output:

-						
<pre>\$ python3 predictive_parser.py</pre>						
Non Terminal	+	*	()	a	\$
E	None	None	E->TG	None	E->TG	None
G	G->+TG	None	None	G->@	None	G->@
T	None	None	T->FU	None	T->FU	None
U	U->@	U->*FU	None	U->@	None	U->@
F	None	None	F->(E)	None	F->a	None