CA10

COMP 141: Parsers

Instructions**:** In this exercise, we are going to review parsers.

1 Recursive-descent parser

1. Consider the following grammar given in EBNF:

$$
\begin{gathered}
expr ::= term{+}term \\
term ::= factor{*}factor \\
factor ::= (expr)|number \\
number ::= NUMBER \\
NUMBER = [0{-}9]+
\end{gathered}
$$

   Give the pseudo-code for recursive-descent parser that implements this grammar.

```
PTNode ParseExpr(){
  PTNode* tree = parseTerm()
  while next_token is '+'
    consume_token()
    tree = new PTInteriorNode('+', tree, parseTerm())
  return tree
}

PTNode parseTerm() {
  PTNode tree = parseFactor()
  while next_token is "*'
    consume_token()
    tree = new PTInteriorNode('*', tree, parseFactor())
  return tree
}

PTNode parseFactor() {
  if next_token is '(' then
    consume_token()
    tree = parseExpr()
    if next_token is ')' then
      consume_token()
      return tree
    else throw exception
  else parseNum()
}

PTNode parseNumber(){
  if next_token is not NUMBER:
    throw exception
  double n = next_token
  consume_token()
  return new PTLeafNode(n)
}
```

2. Consider the following grammar given in EBNF:

$$
\begin{gathered}
expr ::= term[+expr] \\
term ::= factor[*term] \\
factor ::= (expr)|number \\
number ::= NUMBER \\
NUMBER = [0-9]+
\end{gathered}
$$

Give the pseudo-code for recursive-descent parser that implements this grammar.

```
PTNode ParseExpr(){
  PTNode* tree = parseTerm()
  if next_token is '+' then
    consume_token()
    tree = new PTInteriorNode('+', tree, parseTerm())
  return tree
}

PTNode parseTerm() {
  PTNode tree = parseTerm()
  if next_token is "*' then
    consume_token()
    tree = new PTInteriorNode('*', tree, parseFactor())
  return tree
}

PTNode parseFactor() {
  if next_token is '(' then
    consume_token()
    tree = parseExpr()
    if next_token is ')' then
      consume_token()
      return tree
    else throw exception
  else parseNum()
}

PTNode parseNumber(){
  if next_token is not NUMBER:
    throw exception
  double n = next_token
  consume_token()
  return new PTLeafNode(n)
}
```