Question # 4: Give input and output of this mini compiler ?

Ans :**Example 1: Simple Expression**

**Input Expression**: "3 + (4 \* 5) \* 7"

**Steps:**

1. **Lexical Analysis (Lexer)**: The input string "3 + (4 \* 5) \* 7" is tokenized into the following tokens:

**["NUMBER: 3", "PLUS: +", "LPAREN: (", "NUMBER: 4", "TIMES: \*", "NUMBER: 5", "RPAREN: )", "TIMES: \*", "NUMBER: 7"]**

**2.Syntax Parsing (Parser):** The tokens are parsed into a parse tree that represents the structure of the expression:  
+

├── 3

└── \*

├── 4

├── 5

└── 7  
 **Semantic Analysis**: No errors are found in the expression (no division by zero, no mismatched types, etc.).

 **Optimization**: The optimizer simplifies the expression:

* **First, it calculates 4 \* 5 = 20.**
* **Then, it calculates 20 \* 7 = 140.**
* **The tokens are now:**
* **["NUMBER: 3", "PLUS: +", "NUMBER: 140"]**

**Intermediate Code Generation**: The intermediate code (Three-Address Code) is generated:  
t0 = 3

t1 = 20  
t2 = 7  
t3 = t1 \* t2

t4 = t0 + t3

**Code Generation**: The final code is generated:  
t0 = 3

t1 = 20

t2 = 7

t3 = t1 \* t2

t4 = t0 + t3

**Output:**

The generated code for the input expression "3 + (4 \* 5) \* 7" would be:

Generated code for '3 + (4 \* 5) \* 7':

t0 = 3

t1 = 20

t2 = 7

t3 = t1 \* t2

t4 = t0 + t3

A screenshot of a computer

Description automatically generated