**Question 4: how functions works. Step by step**

**1) Lexer:**

**Initialization:**

The Lexer is initialized with the input Brainfork code and a current position pointer.

**Tokenization:**

The lexer iterates through the input code character by character.

It identifies specific characters such as >, <, +, -, ., ,, [, ], and # to create corresponding tokens.

The lexer skips over any invalid characters.

**Token Output:**

For each valid token, the lexer outputs the corresponding Token enum value.

**2) Parser:**

**Initialization:**

The parser is initialized with the lexer and the current token from the lexer.

**Parsing Program:**

The parser starts by parsing the entire Brainfork program.

It iterates through the tokens, recognizing statements and loops.

If a comment token (Token::COMMENT) is encountered, it is skipped.

**Parsing Statements:**

For non-comment tokens, the parser recognizes individual statements (e.g., INC\_PTR, DEC\_PTR, etc.).

Consecutive identical operations are optimized, reducing the number of nodes in the AST.

**Parsing Loops:**

When a LOOP\_START token is encountered, the parser recursively calls the parse Loop function.

Inside a loop, the parser continues to parse statements until a LOOP\_END token is encountered.

**3) Brainfork Compiler:**

**Initialization:**

The Brainfork compiler is initialized with the AST generated by the parser.

**Code Generation:**

The compiler recursively traverses the AST, starting from the root node.

For each node, it generates corresponding Brainfork code based on the type of the token.

The code is printed to the console, creating the final Brainfork code.

**Handling Comments:**

Comment tokens are ignored during code generation.

**Recursion:**

Recursion is employed to handle the hierarchical structure of the AST.

Child nodes are processed within the context of their parent nodes.