COMFORT ABRAHAM
BU/22C/IT/7477

ASSIGNMENT 2

Q1. Build a BNF for a simple Programming language that has support for lexical and syntactic structures like for loops and other loops.

1. Lexical Structure

```
<letter>     ::= a | b | c | ... | z
<digit>      ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<char>       ::= <letter> | <digit> | _
<identifier> ::= <letter> | <identifier><char>
<number>     ::= <digit> | <number><digit>
<operator>   ::= + | - | * | / | % | == | < | > | <= | >=
```

2. Expressions

```
<expression> ::= <number>
            | <identifier>
            | <expression> <operator> <expression>
            | ( <expression> )
```

3. Statements and Control Flow

```
<assignment> ::= <identifier> = <expression> ;

<if_statement> ::= if ( <expression> ) { <block> }
              | if ( <expression> ) { <block> } else { <block> }

<while_loop> ::= while ( <expression> ) { <block> }

<for_loop>   ::= for ( <assignment> <expression> ; <assignment> ) { <block> }

<print_stmt> ::= print ( <expression> ) ;

<statement>  ::= <assignment>
            | <if_statement>
            | <while_loop>
            | <for_loop>
            | <print_stmt>
```

4. Blocks

```
<block> ::= <statement>
        | <block> <statement>
```

5. Test Program: Iterative Factorial

The following sample program calculates the factorial of a number using a for loop. This demonstrates that the BNF grammar supports the required constructs.

```
n = 5;
fact = 1;
for (i = 1; i <= n; i = i + 1) {
    fact = fact * i;
}
print(fact);
```