Lesson Review

Student Name:		Instructor:	
Student No.:	Course / Yr / Blk:	Date:	Score:

TRUE/FALSE: Write **T** if the statement is correct, otherwise **F** if the statement is wrong.

_T__ 1. Syntax refers to the structure or form of a program, while semantics refers to its meaning.

_T__ 2. Syntax and semantics are two fundamental components of any programming language.

_F_ 3. Tokens, keywords, and identifiers are part of the *semantic* structure of a language.

_T__ 4. Formal grammar, such as BNF and EBNF, are used to define the syntax of a programming language.

_T__ 5. Parsing is the process of analyzing the syntax of a program to determine its structure.

_T__ 6. An abstract syntax tree is a representation of the syntactic structure of a program, often used during compilation.

_F__ 7. Operational semantics describes the *meaning* of a program by relating it to the *syntax*.

_T__ 8. Denotational semantics defines the meaning of a program in terms of mathematical functions.

_T__ 9. Axiomatic semantics defines the meaning of a program by specifying the relationships between program statements and logical assertions.

_T__ 10. Lexical analysis is the first phase of compilation, responsible for breaking the source code into tokens.

QUIZ QUESTIONS

Student Name:		Instructor:	
Student No.:	Course / Yr / Blk:	Date:	Score:

MULTIPLE CHOICE: Select and write letter of the best answer on the left side of the number.

_C__1. Which of the following BEST describes the relationship between syntax and semantics in a programming language?

A. Syntax refers to the meaning of a program, while semantics refers to its structure.
B. Syntax and semantics are interchangeable terms.
C. Syntax defines the structure of a program, while semantics defines its meaning.
D. Syntax is concerned with the compiler, while semantics is concerned with the interpreter.

_C__ 2. What are the two primary components of a programming language?

A. Compiler and interpreter
B. Hardware and software
C. Syntax and semantics
D. Variables and data types

_D__ 3. Which of the following is NOT a part of the lexical structure of a programming language?

A. Tokens
B. Keywords
C. Identifiers
D. Abstract Syntax Tree

_B__ 4. What do tokens represent in the lexical structure?

A. The meaning of a program
B. The smallest individual units of a program
C. The overall structure of a program
D. The errors in a program

_C__ 5. What is the purpose of keywords in a programming language?

A. To define variables
B. To perform mathematical operations
C. To serve as reserved words with specific meanings in the language

D. To represent user-defined names

_A__ 6. What are identifiers used for in programming?

A. Naming variables, functions, and other program entities
B. Defining the structure of the language
C. Specifying operations to be performed
D. Representing constant values

_C__ 7. What are BNF and EBNF used for?

A. Defining the semantics of a programming language
B. Specifying the lexical structure of a programming language
C. Describing the formal grammar or syntax of a programming language
D. Implementing a compiler

_C__ 8. What is the role of parsing in the compilation process?

A. To analyze the meaning of a program
B. To break down the program's source code into tokens
C. To construct an Abstract Syntax Tree (AST) representing the program's structure
D. To generate machine code

_D__ 9. Which of the following is NOT a common approach to defining the semantics of a programming language?

A. Operational semantics
B. Denotational semantics
C. Axiomatic semantics
D. Lexical semantics

_C__ 10. Which approach to semantics uses logical axioms and rules to specify program behavior?

A. Operational semantics
B. Denotational semantics
C. Axiomatic semantics
D. Lexical semantics

Laboratory Exercises

Student Name:		Instructor:	
Student No.:	Course / Yr / Blk:	Date:	Score:

Skills Challenge

1. **Identifying Tokens and Lexemes**

   Objective: To understand the lexical structure of a programming language and identify different token types.

   Task: Analyze the following code snippet, you can choose a language such as Python or Java

```
x = 10;
y = "hello";
  if (x > 5):
    print(y);
```

   Identify and list all the tokens present in the code, classifying them into categories like identifiers, keywords, operators, literals, and delimiters. For each token, specify its lexeme

	Token	Lexem
Identifier	x	x
Operator	=	=
Literal	1	1
	0	0

Delimiter	;	;
Identifier	y	y
Literal	"	"
	h	h
	e	e
	l	l
	l	l
	o	o
	"	"
Keyword	i	i
	f	f
Delimiter	(	(
Identifier	x	x
Operator	>	>
Literal	5	5
Delimiter	)	)
Delimiter	:	:
Identifier	p	p
	r	r

	i	i
	n	n
	t	t
Delimiter	(	(
Identifier	y	y
Delimiter	)	)
Delimiter	;	;

2. **BNF/EBNF Grammar for Simple Expressions**

Objective: To learn how to define the syntax of a simple language using formal grammars.

Task: Develop a BNF or EBNF grammar for arithmetic expressions involving addition, subtraction, multiplication, and division, along with integer literals and single-letter variables. Consider operator precedence.

<expr> ::= <term> { ("+" | "-") <term> }
<term> ::= <factor> { ("*" | "/") <factor> }
<factor> ::= <number> | <variable> | "(" <expr> ")"
<number> ::= digit { digit }
<variable> ::= letter
digit ::= "0" | "1" | ... | "9"
letter ::= "a" | "b" | ... | "z" | "A" | ... | "Z"

3. **Constructing Abstract Syntax Trees (ASTs)**
   - **Objective:** To understand the process of parsing and representing code in a tree structure.
   - **Task:** Draw the Abstract Syntax Tree (AST) for the following expression, based on the gramma