1. Define the following: syntax, semantics, sentence, language, lexeme, token.

* Syntax: The form or structure of the expressions, statements, and program units
* Semantics: The meaning of the expressions, statements and program units.
* Sentence: A sentence is a string of characters over some alphabet.
* Language: A language is a set of sentences
* Lexeme: A lexeme is the lowest level syntactic unit of a language(e.g., \*, sum, begin)
* Token: A token is a category of lexemes (e.g., identifier)

1. Who are language descriptions for?

* **Language descriptions are for** 
  + **Other language designers**
  + **Implementers**
  + **Programmers (the users of the language)**

1. Describe the operation of a general language generator.

* **A device that generates sentences of a language**
* **One can determine if the syntax of a particular sentence is syntactically correct by comparing it to the structure of the generator.**

1. Describe the operation of a general language recognizer.

* **A recognition device reads input strings over the alphabet of the language and decides whether the input strings belong to the language**
* **Example: Syntax analysis part of a compiler**
  + **Detailed discussion of syntax analysis appears in chapter 4.**

1. What is the difference between a sentence and a sentential form?

* Every string of symbols in a derivation is a sentential form
* A sentence is a sentential form that has only terminal symbols

1. What is BNF?

* BNF is a metalanguage invented by John Backus to describe the syntax of Algol 58. It is equivalent to context-free grammars.

1. What is a leftmost derivation?

* A leftmost derivation is one in which the leftmost nonterminal in each sentential form is the one that is expanded.

1. What is a parse tree?

* One of the most attractive features of grammars is that they naturally describe the hierarchical syntactic structure of the sentences of the languages they define. These hierarchical structures are called parse trees.

1. What is an ambiguous grammar?

* A grammar that generates a sentential form for which there are two or more parse trees can be said to be ambiguous.

1. How can ambiguous grammar be made unambiguous?

* Changing the grammar

1. How is precedence indicated in a grammar?

* ***When an expression has two operators that have the same precedence, a semantic rule is required to specify which should have precedence. This rule is called associativity.***

1. How is a left-recursive grammar rule defined?

* ***When a grammar rule has its LHS appearing at the beginning of its RHS, it is said to be left recursive. This left recursion specifies left associativity***

1. How is right-associativity indicated in a grammar?

* ***In most cases, the exponentiation operator is right associative. Right recursion can be used to indicate right associativity. Right recursion is indicated by putting the LHS at the end of the RHS***

1. What three extensions are common to most EBNFs?

* Brackets, braces and parentheses. Three extensions are commonly used in:

1. Optional parts are placed in brackets[]

<proc \_call> 🡪 ident[(<expr\_lilst>)]

1. Alternative parts of RHSs are placed inside parentheses and separated via vertical bars

<term> 🡪 <term> (+|-) const

1. Repetitions (0 or more) are placed inside braces {}

<ident> 🡪 letter {letter|digit}

1. What is a metasymbol?

* EBNF uses brackets, braces and parentheses as notational tools. These notational tools are metasmbols. They are not terminal symbols in syntactic entities that they describe.

1. What are static semantics?

* Static semantics are language rules that are only indirectly related to the meaning of the programs at execution. They have to do with the legal forms of programs at compile time. Most static semantics rules have to do with type constraints.

1. What is an attribute grammar?

* Attribute grammars (AGs) are a formal approach to both describing and checking the correctness of static semantic rules of a program.

1. What is the primary use of attribute grammars?

* Static semantics specifications
* Compiler design (static semantics checking)

1. What is the difference between a synthesized and an inherited attribute?

* Synthesized attributes are used to pass semantic information up a parse tree.
* Inherited attributes pass semantic information down and across a parse tree.

1. How is the process of computing the attributes determined for the trees of a given attribute grammar?

* *The process of computing the attribute values of a parse tree, is sometimes called* ***decorating*** *the parse tree. If all attributes were inherited, this could proceed in a completely top-down order, from the root to the leaves*

1. Evaluate attribute grammars.

* *Checking the static semantic rules of a language is an essential part of all compilers.*
* *One of the main difficulties in using an AG to describe all of the syntax and static semantics of a real contemporary programming language is the size and complexity of the resulting AG.*
* *The large number of attributes and semantic rules required for a complete programming language make such grammars difficult to write and read. Furthermore, the attribute values on a large parse tree are costly to evaluate.*
* *On the other hand, less formal AGs are a powerful and commonly used tool for compiler writers, who are more interested in the process of producing a compiler than they are in formalism.*