NAME: **IBEGBU EJIAMIKE ALBERT**

MATRIC NO: **17/SCI01/036**

COURSE CODE: **CSC 312**

**ASSIGNMENT**

**ANSWER**

A grammar is a powerful tool for describing and analyzing languages. It is a set of rules by which valid sentences in a language are constructed. A grammar (when the context is not given, often called a formal grammar for clarity) describes how to form strings from a language's [alphabet](https://en.wikipedia.org/wiki/Alphabet_%28computer_science%29%22%20%5Co%20%22Alphabet%20%28computer%20science%29) that are valid according to the language's [syntax](https://en.wikipedia.org/wiki/Syntax_%28programming_languages%29%22%20%5Co%20%22Syntax%20%28programming%20languages%29). A grammar does not describe the [meaning of the strings](https://en.wikipedia.org/wiki/Semantics%22%20%5Co%20%22Semantics) or what can be done with them in whatever context—only their form. A formal grammar is defined as a set of [production rules](https://en.wikipedia.org/wiki/Production_%28computer_science%29%22%20%5Co%20%22Production%20%28computer%20science%29) for [strings](https://en.wikipedia.org/wiki/String_%28computer_science%29%22%20%5Co%20%22String%20%28computer%20science%29) in a formal language. A formal grammar is a set of rules for rewriting strings, along with a "start symbol" from which rewriting starts. Therefore, a grammar is usually thought of as a language generator.

1. **DERIVATION**

A sequence of applications of the rules of a grammar that produces a finished string of terminals. A leftmost derivation is where we always substitute for the leftmost non-terminal as we apply the rules (we can similarly define a rightmost derivation). A derivation is also called a parse. Strings may be derived from other strings using the productions in a grammar. If a grammar **G** has a production **α → β**, we can say that **x α y** derives **x β y** in **G**. This derivation is written as −

 ***x α y ⇒G x β y***

1. **PRODUCTION**

A grammar rule that describes how to replace/exchange symbols. The general form of a production for a non-terminal is: X –>Y1Y2Y3...Yn. The non-terminal X is declared equivalent to the concatenation of the symbols Y1Y2Y3...Yn. The production means that anywhere where we encounter X, we may replace it by the string Y1Y2Y3...Yn. Eventually we will have a string containing nothing that can be expanded further, i.e., it will consist of only terminals. Such a string is called a sentence. In the context of programming languages, a sentence is a syntactically correct and complete program.

1. **SENTENCE**

A sentence is a string of symbols in T derived from S using one or more applications of productions in P. A string of symbols derived from S but possibly including non-terminals is called a sentential form or a working string.

1. **NULL SYMBOL ε**

It is sometimes useful to specify that a symbol can be replaced by nothing at all. To indicate this, we use the null symbol ε, e.g., A –> B |ε.