1. List various types of compilers.

There are three types of compilers are described below:

* Single-Pass Compilers
* Two-Pass Compilers
* Multipass Compilers

1. Differentiate Tokens, Patterns, and Lexeme.

* **Tokens:** Tokens are character sequences that have significance when taken together as a whole.
* **Patterns:** Patterns are recurring occurrences of the exact string in the input that result in the generation of the same token in the output. The rule referred to as a pattern is associated with the token and is used to characterize this group of strings.
* **Lexeme:** It is a string of characters in the source code used to determine whether or not a token should be granted access. The fundamental units of any language are called tokens.

1. What are the six phases of a compiler?

The 6 phases of a compiler are:

* Syntactic Analysis or Parsing.
* Intermediate Code Generation.
* Lexical Analysis.
* Code Optimization.
* Code Generation.
* Semantic Analysis.

### Can you explain context-free grammar and its importance in compiler design?

Grammar is a set of rules that specify how a language might be formed. A context-free grammar is a type of grammar. It is an important point in the design of compilers since the compiler needs to comprehend the structure of the programming language it is translating into machine code to do it accurately.

A blue rectangle with white text

Description automatically generated

### What is lexical analysis?

The technique of determining which lexemes are present in a sentence is known as lexical analysis. Words and morphemes are common names for lexemes, the more fundamental units of meaning in a language. Lexemes are also sometimes referred to as morphemes. Lexical analysis is used not just for studying written texts and phrases spoken aloud. However, it also has applications for analyzing the spoken language used in naturalistic research.

A blue rectangle with white text

Description automatically generated

6.What are compilers and interpreters?

* **Compiler:** A compiler translates the complete source code in a single run. A compiler is more efficient than an interpreter since it requires less time to complete the task.
* **Interpreters:** An interpreter performs a line-by-line translation of the complete source code. The interpreter requires much more time than the compiler, which means it is considerably slower than the compiler.

A diagram of a machine

Description automatically generated

### 7.What is the fastest compiler language?

C++ is an easy and efficient programming language. Its fast runtime and extensive collection of Standard Template Libraries make it a favorite among competitive programmers (STL).

### 8.What are some examples of compile-time and runtime errors?

Any fault during the software compilation process is a compile-time error. Any error that occurs while a program is being executed is referred to as a runtime error.

**Examples**: Syntax mistakes, type errors, and name errors are all examples of errors that might occur during the compilation process. Error during the runtime includes illegal type conversion, division by zero, and indexing outside the allowed range.

### 9. What are some examples of compile-time and runtime errors?

Any fault during the software compilation process is a compile-time error. Any error that occurs while a program is being executed is referred to as a runtime error.

**Examples**: Syntax mistakes, type errors, and name errors are all examples of errors that might occur during the compilation process. Error during the runtime includes illegal type conversion, division by zero, and indexing outside the allowed range.

10. Briefly explain parts of the compilation.

The compiler consists of several phases:  
• Lexical Analysis: This phase is also known as scanning. The source code is broken down into a sequence of tokens.  
• Syntax Analysis: This phase checks the grammar of the code. A parser reads the token generated by the scanner and builds an abstract syntax tree.  
• Semantic Analysis: It is also known as context analysis. The semantic analyzer reads the AST produced during syntax analysis. Then it performs type checking, scope analysis, and other error.  
• Code Optimization: This involves improving the final code, which is done by reducing file size, increasing speed, reducing power consumption, etc.  
• Code Generation: It involves generating machine code from the intermediate after all the phases.

11. What are code optimisation techniques?

### Different Types of Optimization

Optimization is classified broadly into two types:

* Machine-Independent
* Machine-Dependent

## **Machine-Independent Optimization**

It positively affects the efficiency of intermediate code by transforming a part of code that does not employ hardware parts. It usually optimises code by eliminating tediums and removing unneeded code.

do

{

item = 10;

amount= amount + item;

} while(amount<100);

This code implicates the replicated assignment of the identifier item if we put it this way:

item = 10;

do

{

amount= amount + item;

} while(amount<100);

## **Machine-Independent Optimization Techniques:**

* Compile Time Evaluation
* Common Subexpression Elimination
* Variable Propagation
* Dead Code Elimination
* Code Movement
* Strength Reduction

## **Machine-Dependent Optimization**

After the target code has been constructed and transformed according to the target machine architecture, machine-dependent optimization is performed. It makes use of CPU registers and may utilise absolute rather than relative memory addresses. Machine-dependent optimizers work hard to maximise the perks of the memory hierarchy.

# 12. What is Context-Free Grammar?

Context Free Grammar is formal grammar, the syntax or structure of a formal language can be described using context-free grammar (CFG), a type of formal grammar. The grammar has four tuples: (V,T,P,S).

V - It is the collection of variables or nonterminal symbols.  
T - It is a set of terminals.   
P - It is the production rules that consist of both terminals and nonterminals.  
S - It is the Starting symbol.