Name: Arslan Ahmed

Reg#: Fa21-bcs-026

Subject: CC Lab Terminal

Question 5: Explain the function which performs semantic analysis in your mini compiler?

In a mini compiler, semantic analysis is the phase where the compiler ensures that the program makes sense in terms of the language's rules and logic. This step follows syntax analysis (parsing) and involves checking for various types of errors that are logically invalid but do not necessarily break the grammar.

In a typical mini compiler written in C#, semantic analysis might involve the following major tasks:

**Symbol Table Construction:** The symbol table stores information about variables, functions, classes, and other program entities. During semantic analysis, the compiler will check the declaration of each identifier, ensuring that it has been defined before it is used.

* **Scope Management:** Ensuring that variables are used in the correct scope. For instance, a variable declared in a function should not be accessible outside it unless it is returned or passed to another function.
* **Type Checking:** Ensuring that operations are performed on compatible types. For example, attempting to add an integer to a string is semantically incorrect in many languages.

**Type Checking:** The compiler checks whether operations in the code make sense with respect to the data types. For example, adding a string to an integer or calling a function with the wrong type of arguments would generate an error.

* **Type consistency:** Making sure that variables, expressions, and function calls have compatible types.
* **Return type checking:** Ensuring that functions return values of the correct type.
* **Operator type compatibility:** Ensuring operators are used correctly (e.g., trying to divide by zero).

**Scope Checking:**

The compiler checks if variables are declared in the correct scope and are accessible. If a variable is used before being declared, or if it is out of scope, an error will occur.

**Control Flow Analysis:**

Ensuring the correct use of control flow structures, like if, while, for, and switch. For example, the compiler may ensure that all paths in a function return a value when necessary (like ensuring a return statement exists for a non-void function).

**Function and Method Call Checking:**

Ensuring that the correct number and types of arguments are passed to functions or methods.

Verifying that the return type of a function matches the expected type in the call.

**Array Bound Checking:**

Ensuring that indices for arrays are within their valid bounds.