Logo, company name

Description automatically generated

**COMSATS University Islamabad (CUI)**

**Lab Terminal**

**Submitted to: Sir Bilal Bukhari**

**Submitted By: Sohaib Safeer**

**Reg No: FA21-BCS-054**

**Course: Topics in Computer Science**

**Date: 3 jan, 2025**

**Question 6 : explain symentic analysis in your mini compiler**

**Semantic Analysis in a Mini Compiler**

In a mini compiler, semantic analysis is a crucial phase that checks the source code for meaning and consistency beyond the syntactic structure. It ensures that the program adheres to the language's rules and makes sense within the context of the program.

**Key Tasks of Semantic Analysis:**

1. **Type Checking:**
   * **Verify Type Compatibility:** Ensures that operands of operators have compatible data types. For example, it checks that you're not adding an integer to a string.
   * **Type Inference:** Determines the types of expressions and variables based on the context and declared types.
   * **Coercion (Implicit Type Conversion):** In some cases, the compiler might automatically convert one data type to another (e.g., converting an integer to a floating-point number for an operation).
2. **Scope Resolution:**
   * **Determine Variable Scope:** Identifies the scope (global, local) of each variable and ensures that variables are used within their declared scope.
   * **Resolve Name Conflicts:** Checks for duplicate variable names within the same scope and handles them appropriately.
3. **Checking for Undeclared Identifiers:**
   * Verifies that all variables and functions used in the program have been declared before their use.
4. **Checking for Function Calls:**
   * Ensures that function calls have the correct number and types of arguments.

**Example:**

Consider the following code snippet:

C

int x = 5;

char y = 'a';

int z = x + y;

During semantic analysis:

* The compiler would detect a type mismatch in the expression x + y.
* It would report an error, as you cannot directly add an integer (x) to a character (y).

**Implementation in the Mini Compiler:**

* **Symbol Table:** The semantic analyzer heavily relies on the symbol table to store information about variables, their types, and their scopes.
* **Type Checking Routines:** The compiler would have functions to check the compatibility of operands for different operators, such as check\_addition\_types(), check\_assignment\_types(), etc.
* **Scope Management:** The compiler would maintain a stack of scopes to keep track of the current scope and resolve variable references correctly.

By performing these checks, semantic analysis helps catch errors that might not be detected by the syntax analyzer alone, leading to more robust and correct programs.

**Note:** The specific implementation of semantic analysis can vary depending on the complexity of the language and the design of the mini compiler.