|  |  |
| --- | --- |
| **https://lh7-us.googleusercontent.com/cqr1zywKLZ-KYtGHQsJs_4r0Pz65g7Hm9cYAB_QVFrqm6JK4FqjMzqYGew6RHuFzmfT56Wdn2C69ISfHmuDwuy_tnadpQXO2ujqRH_tBzkPaOHrK6awj4voQaDIwnQBQeNfh8u1bi15aREV24NpZmw** | **Compiler Construction**  **BSCS 5-A**  **Department of Computer Science**  **Bahria University, Lahore Campus** |

**Assignment: [1]**

Date: Week 2, March 2024

Name: \_AFFAN AHMAD\_\_\_\_\_\_\_

Roll No:  \_03-134221-003\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation of CLO** | **Question Number** | **Marks** | **Obtained Marks** |
| **CLO1:  Understand the basic techniques used in compiler construction.** | 1 | 5 |  |
|  |  |  |
| **Total Marks** | | **5** |  |

**Q1: Consider set of instruction involved in computation of Fibonacci sequence.**

1. **Pass these instructions into each phase of compiler and provide optimized code instructions as final output.**
2. **Explicitly mention input and output of each phase.**

**Avoid plagiarism.**

**Program code is supposed to be written individually , so similarity in code will reflect in similarity of input and output of each phase (deliverable). That will lead to 50% deduction in obtained marks.**

**Lexical Analysis:**

**Input: Set of source code instructions for computing the Fibonacci sequence.**

**Output: Tokens identified in the source code, such as identifiers, keywords, literals, and punctuation.**

**Example: int fib(int n) { return (n <= 1) ? n : fib(n - 1) + fib(n - 2); }**

**Syntax Analysis:**

**Input: Tokens from the lexical analysis phase.**

**Output: Abstract syntax tree (AST) representing the syntactic structure of the program.**

**Example: AST representation of the Fibonacci function.**

**Semantic Analysis:**

**Input: AST from the syntax analysis phase.**

**Output: Identification and resolution of semantic errors, type checking, and symbol table construction.**

**Example: Checking for type consistency, ensuring that the Fibonacci function is correctly defined and used.**

**Intermediate Code Generation:**

**Input: Semantically analyzed AST.**

**Output: Intermediate representation (IR) code that is closer to machine language but still platform-independent.**

**Example: IR code representing the Fibonacci function, such as three-address code or LLVM IR.**

**Optimization:**

**Input: Intermediate code representation.**

**Output: Optimized code with improved performance, reduced memory usage, or minimized execution time.**

**Example: Apply optimization techniques like constant folding, loop unrolling, or tail recursion elimination to improve the efficiency of the Fibonacci computation.**

**Code Generation:**

**Input: Optimized intermediate code.**

**Output: Machine code or assembly code specific to the target architecture.**

**Example: Generated machine code or assembly instructions for the Fibonacci function, ready to be executed on the target platform.**

Optimized Code Instructions (Final Output):

int fib(int n) {

if (n <= 1)

return n;

int a = 0, b = 1, next;

for (int i = 2; i <= n; ++i) {

next = a + b;

a = b;

b = next;

}

return b;

}