|  |  |
| --- | --- |
|  | **COMSATS University Islamabad, Attock Campus**  **Lab Terminal Examinations (Spring 2024)** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Department of: | | **Computer Science** | | | | |  |
|  | |  | |  | | | | |  |
| Class/Program: | | **BS(CS)-7th** | | Date: | **31/05/ 2024 (1:30 - 4:30)** | | | | |
| Subject: | **Compiler construction Lab EXAM** | | | Instructor: | | **Bilal Haider** | | | |
| Total Time Allowed: | | | **3Hrs** | Maximum Marks: | | | | **50** | |
| Student Name: | | Neha Zainab | | Registration #: | | | Sp21-bcs-024 | | |
|  | |  | |  | | |  | | |

**To submit create a world file titled csc441-sp23-lab terminal-your complete registration number.docx.**

**Each question has 10 marks, please make a word file for answer of question 1 and 2, add screen shots of your input and output of question 3 and 4 in the same world document. Upload your code for project, question3 and question on google drive and paste it at the end of the word document**

**Viva will have 10 marks you can give viva until 27th of june 2023.**

**Question 1**

Write an introduction of your compiler construction project

**Answer:**

**Introduction:**

Compiler construction is a fundamental aspect of computer science that bridges the gap between high-level programming languages and machine code that can be executed by a computer's hardware. Our project, the Mini-Python Compiler, serves as an educational tool to demonstrate the essential phases of compiler construction, specifically for a subset of the Python programming language.

The Mini-Python Compiler project is designed to translate a simplified version of Python code into intermediate code and then into a hypothetical assembly-like language. This process involves several key stages, each critical for converting human-readable code into low-level instructions. The primary components of our compiler include lexical analysis, syntax analysis, semantic analysis, intermediate code generation, and target code generation.

For this mini-compiler, the following aspects of the Python language syntax have been covered

* Constructs like ‘if-else’ and ‘while’ and the required indentation for these loops.
* Nested loops
* Integer and float data types

Specific error messages are displayed based on the type of error. Syntax errors are handled using the yyerror() function, while the semantic errors are handled by making a call to a function that searches for a particular identifier in the symbol table. The line number is displayed as part of the error message.

As a part of error recovery, panic mode recovery has been implemented for the lexer. It recovers from errors in variable declaration. In case of identifiers, when the name begins with a digit, the compiler neglects the digit and considers the rest as the identifier name. Languages used to develop this project:

* C
* YACC
* LEX
* PYTHON

**DIFFERENT MODULES OF PROJECT**

**Token And Symbol Table:**

This folder contains the code that outputs the tokens and the symbol table.

**Abstract Syntax Tree:**

This folder contains the code that displays the abstract syntax tree.

**Intermediate Code Generation:**

This folder contains the code that generates the symbol table before optimisations and the intermediate code.

**Optimized ICG:**

This folder contains the code that generates the symbol table after optimizations, the quadruples table and the optimized intermediate code.

**Target Code:**

This folder contains the code that displays the assembly code/target code

**Different Files:**

**proj.l:**

It is the Lexical analyser file which defines all the terminals of the productions stated in the yacc file. It contains regular expressions.

**proj1.y:**

Yacc file is where the productions for the conditional statements like if-else and while and expressions are mentioned. This file also contains the semantic rules defined against every production necessary. Rules for producing three address code is also present.

**final.py:**

It is the python file which converts the ICG to target code using regex.

**inp.py:**

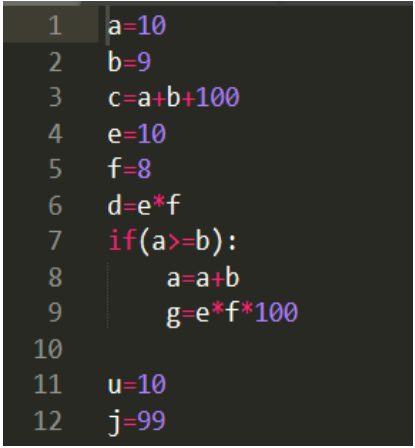
The input python code which will be parsed and checked for semantic correctness by executing the lex and yacc files along with it.

**Question 2**

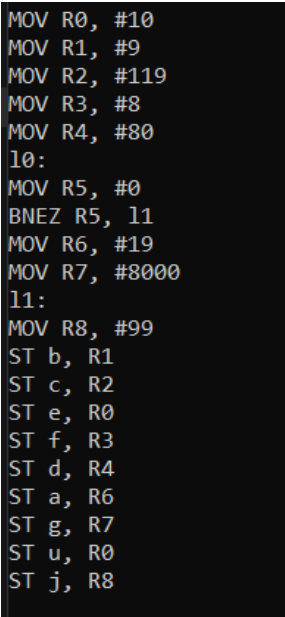
Give a sample input and output for your compiler construction project

**Anwser:**

**Sample input:**

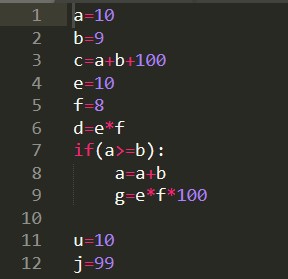


**Output:**

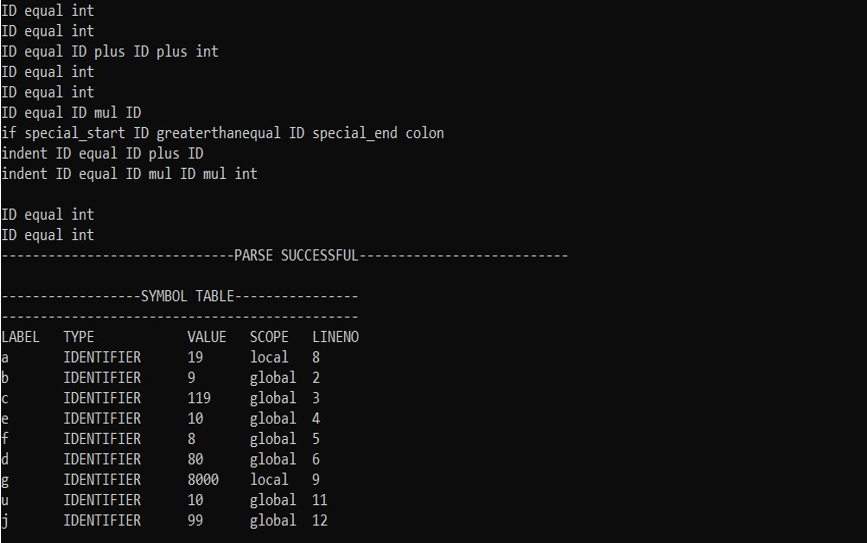


**How the process being:**

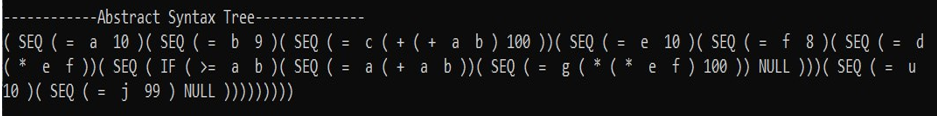
**Input:**



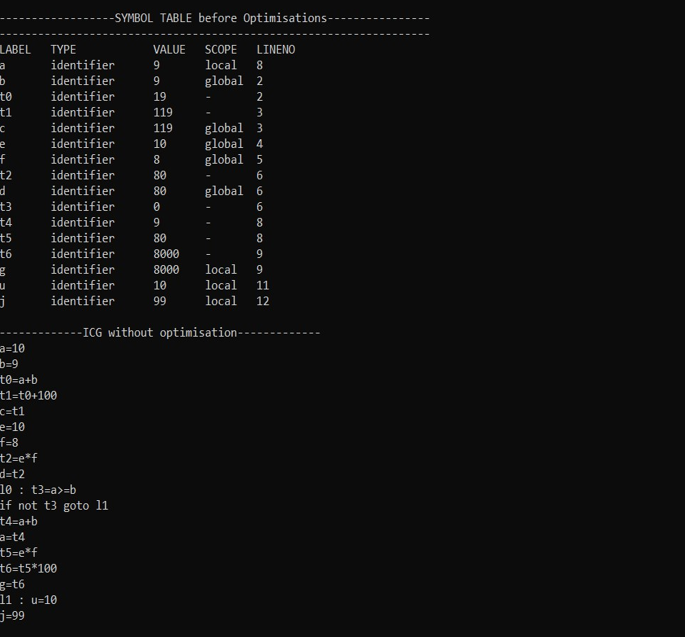
**Tokens and Symbol Table:**



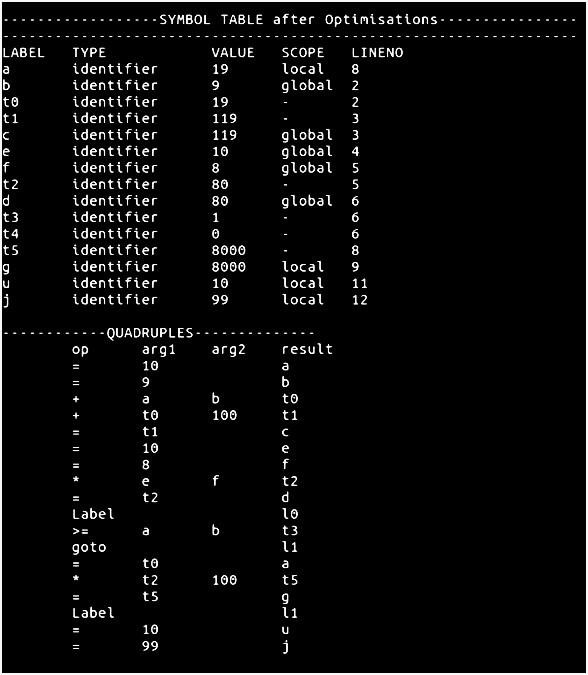
**Abstract Syntax Tree:**

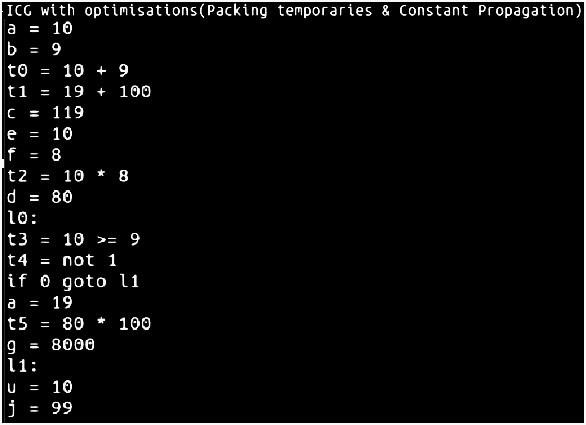


**Symbol Table and Unoptimized Intermediate Code:**

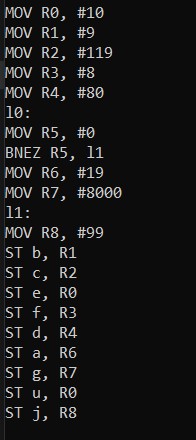


**Symbol Table, Quadruples Table and Optimized Intermediate Code:**



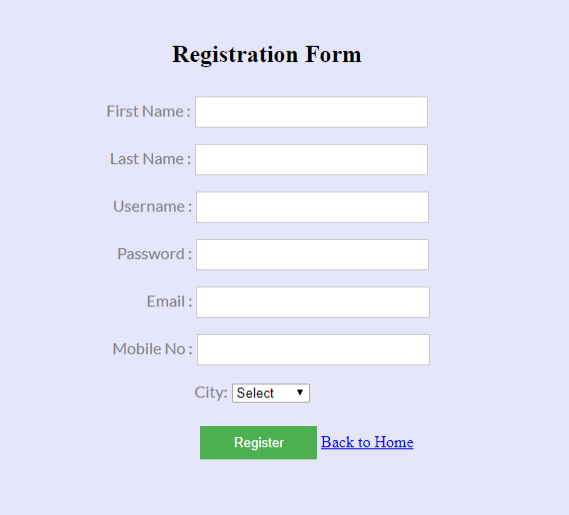


**Target Code:**



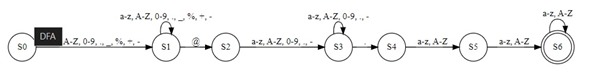
**Question 3**

Create and implement RE and DFAs for the form below



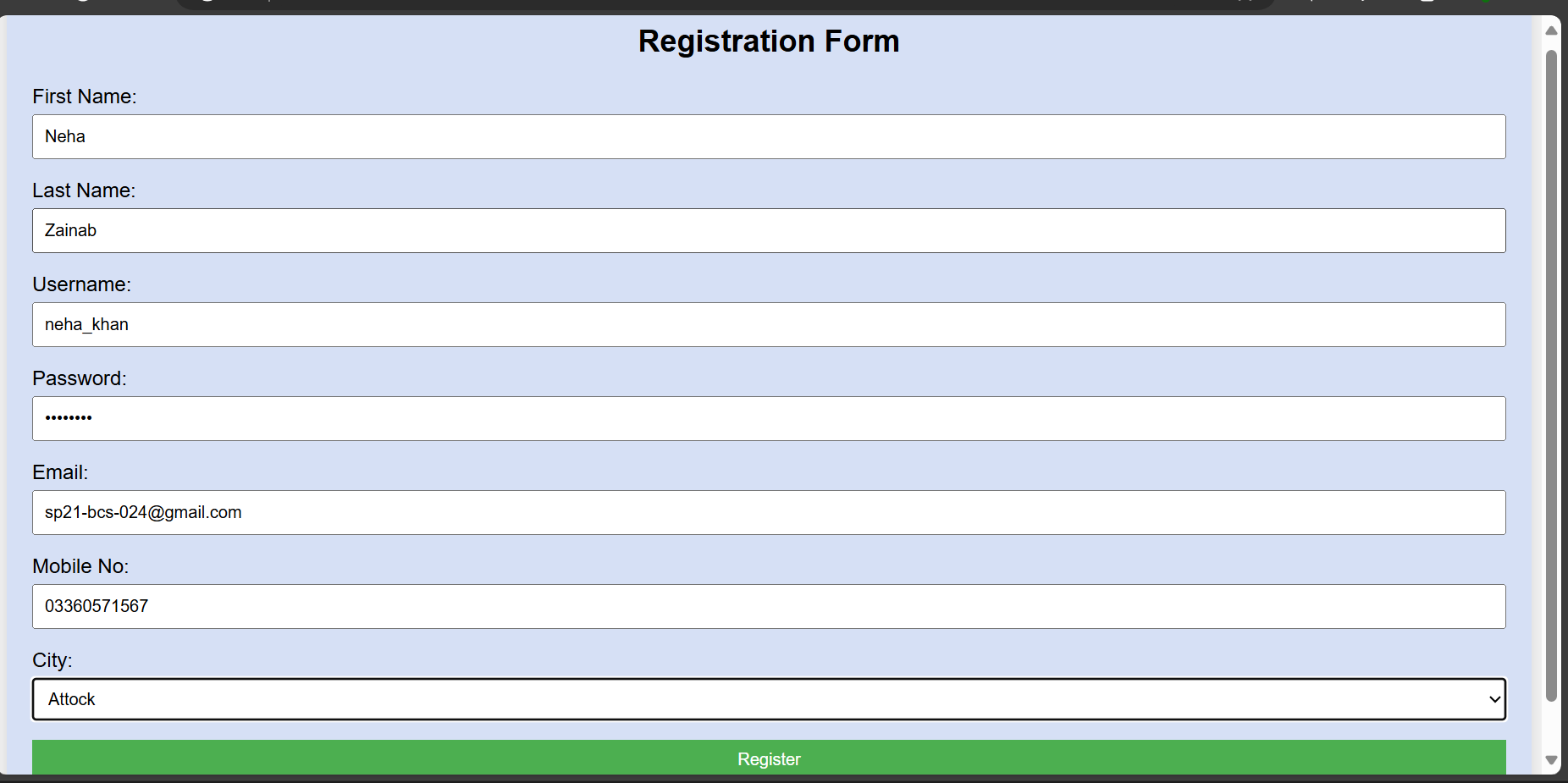
You must use Regex to validate data.

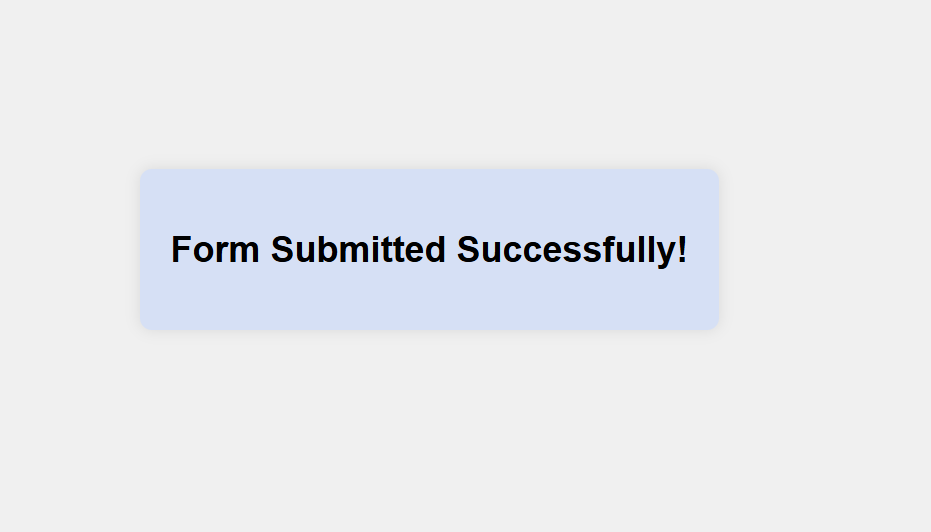
**Answer:**

****

**Question 4:**

Write a program which generates symbol table for the code you submitted in question 3

****

****