|  |  |
| --- | --- |
|  | **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: | | SYED ABDULLAH  MUHAMMAD HASEEB | | Registration #: | | | Sp21-BCS-037  SP21-BCS-020 | | |
|  | |  | |  | | |  | | |

**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.**

**Question 1**

Write an introduction of your compiler construction project

**INTRODUCTION:**

We have built a self-compiling mini-compiler for C++. C++ is a general purpose programming language and widely used now a days for competitive programming. It has imperative, object-oriented and generic programming features. The two main constructs that we have focused on while building this compiler are ‘while’ and ‘for’ loops. The compiler also identifies arithmetic, Boolean and logical operations. The expected outcome of this project is generate a Symbol Table, an Abstract Syntax Tree and Intermediate Three-Address code along with optimization.

**Question 2**

Give a sample input and output for your compiler construction project.

**INPUT:**

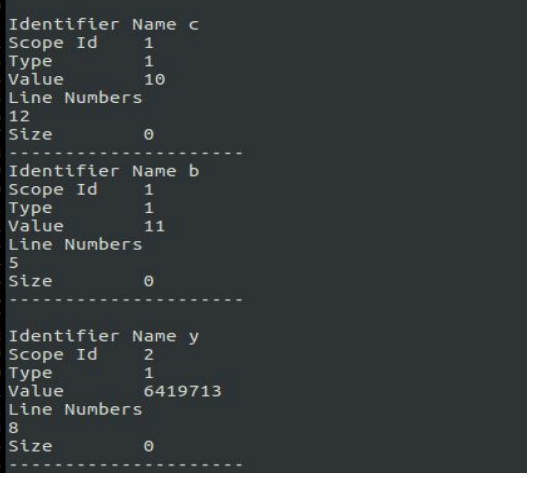
****

**OUTPUT:**

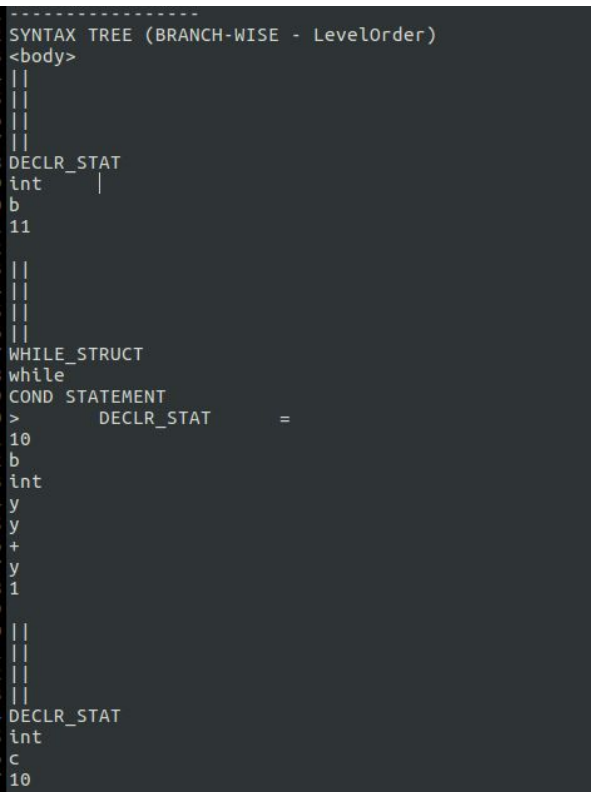
**A screenshot of a computer

Description automatically generated**

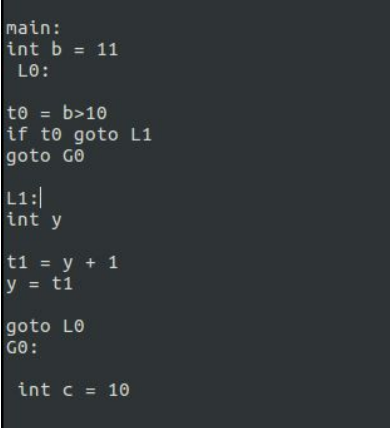
**SYMBOL TABLE**

****

**ABSTRACT SYNTAX TREE**

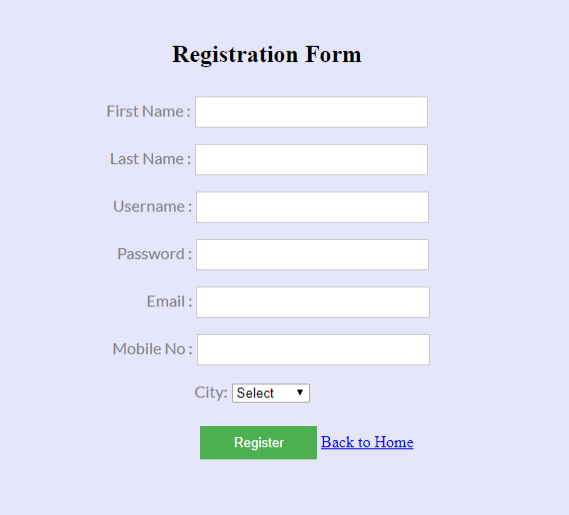
****

**INTERMEDIATE CODE GENERATION**

****

**Question 3**

Create and implement RE and DFAs for the form below



You must use Regex to validate data.

**ANSWER:**

import re

# Regular Expressions

regexes = {

"first\_name": re.compile(r"^[A-Za-z]+$"),

"last\_name": re.compile(r"^[A-Za-z]+$"),

"username": re.compile(r"^[A-Za-z0-9\_]{3,16}$"),

"password": re.compile(r"^(?=.[A-Za-z])(?=.\d)(?=.[@$!%?&])[A-Za-z\d@$!%\*?&]{8,}$"),

"email": re.compile(r"^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"),

"mobile\_no": re.compile(r"^\d{10}$")

}

# DFA Simulation Function

def validate\_input(field, value):

if field in regexes:

return bool(regexes[field].match(value))

return False

# Test Cases

test\_data = {

"first\_name": "John",

"last\_name": "Doe",

"username": "john\_doe\_123",

"password": "Passw0rd!",

"email": "john.doe@example.com",

"mobile\_no": "1234567890"

}

# Validation

validation\_results = {field: validate\_input(field, value) for field, value in test\_data.items()}

print(validation\_results)

**Question 4:**

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

class SymbolTable:

def \_init\_(self):

self.table = {}

def add(self, name, type, value=None):

self.table[name] = {"type": type, "value": value}

def get(self, name):

return self.table.get(name, None)

def \_str\_(self):

result = "Symbol Table:\n"

result += "Name\t\tType\t\tValue\n"

result += "-"\*40 + "\n"

for name, info in self.table.items():

result += f"{name}\t\t{info['type']}\t\t{info['value']}\n"

return result

# Creating the symbol table

symbol\_table = SymbolTable()

# Adding variables to the symbol table

symbol\_table.add("regexes", "dictionary", {

"first\_name": r"^[A-Za-z]+$",

"last\_name": r"^[A-Za-z]+$",

"username": r"^[A-Za-z0-9\_]{3,16}$",

"password": r"^(?=.[A-Za-z])(?=.\d)(?=.[@$!%?&])[A-Za-z\d@$!%\*?&]{8,}$",

"email": r"^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$",

"mobile\_no": r"^\d{10}$"

})

symbol\_table.add("validate\_input", "function", "validates input based on regexes")

symbol\_table.add("test\_data", "dictionary", {

"first\_name": "John",

"last\_name": "Doe",

"username": "john\_doe\_123",

"password": "Passw0rd!",

"email": "john.doe@example.com",

"mobile\_no": "1234567890"

})

symbol\_table.add("validation\_results", "dictionary", None)

regexes = {

"first\_name": re.compile(r"^[A-Za-z]+$"),

"last\_name": re.compile(r"^[A-Za-z]+$"),

"username": re.compile(r"^[A-Za-z0-9\_]{3,16}$"),

"password": re.compile(r"^(?=.[A-Za-z])(?=.\d)(?=.[@$!%?&])[A-Za-z\d@$!%\*?&]{8,}$"),

"email": re.compile(r"^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"),

"mobile\_no": re.compile(r"^\d{10}$")

}

# Adding regexes to the symbol table

for key in regexes:

symbol\_table.add(f"regex\_{key}","regex", regexes[key].pattern)

# Printing the symbol table

print(symbol\_table)