|  |  |
| --- | --- |
|  | **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: | | **SAFWANA….**  **SAVEEZA……** | | Registration #: | | | **SP21-BCS-028**  **SP21-BCS-034** | | |
|  | |  | |  | | |  | | |

**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**

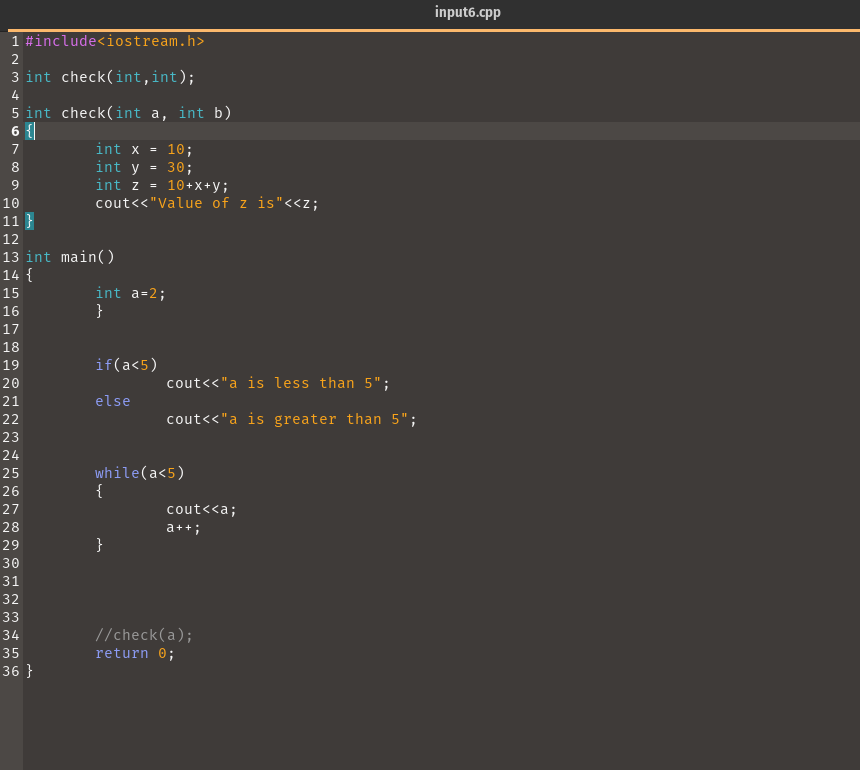
**Write an introduction of your compiler construction project**

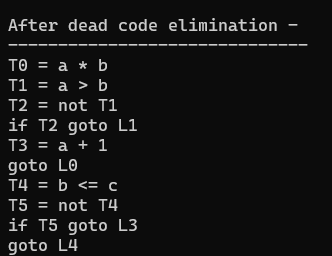
The goal of our compiler construction project is to develop a fully functional compiler for a simple programming language. This compiler will translate high-level code into machine code or an intermediate form that can be executed by a virtual machine. The project covers several critical phases of compiler design, including lexical analysis, syntax analysis, semantic analysis, optimization, and code generation.

Our compiler will start by converting source code into tokens using a lexical analyzer. These tokens will then be parsed into a syntactic structure using a parser, ensuring that the code follows the language's grammar rules. Next, a semantic analyzer will verify that the program adheres to semantic rules, such as type checking. Finally, the compiler will optimize the code and generate executable machine code or bytecode.

**Question 2**

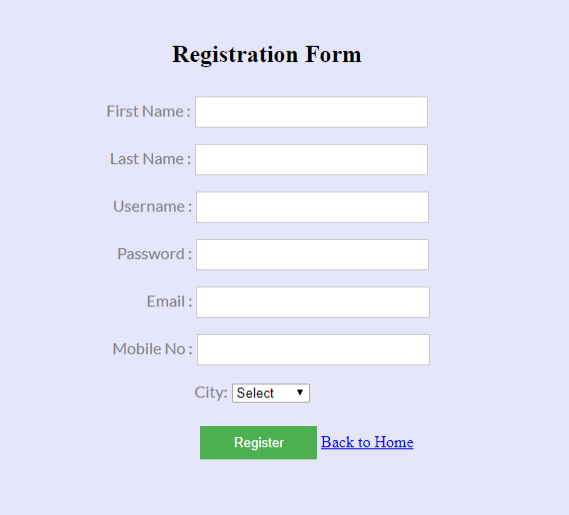
**Give a sample input and output for your compiler construction project**





**Question 3**

**Create and implement RE and DFAs for the form below**

****

**You must use Regex to validate data.**

**Re and dfa:**

#include <iostream>

#include <regex>

#include <vector>

#include <string>

using namespace std;

// Function prototypes

bool validate\_regex(const string& str, const string& pattern);

bool validate\_city(const string& city);

int main() {

// Sample data for testing

string first\_name = "John";

string last\_name = "Doe";

string username = "johndoe123";

string password = "password123";

string email = "john.doe@example.com";

string mobile\_number = "1234567890";

string city = "New York";

// Regular expressions for validation

const string name\_pattern = "^[A-Za-z]+$";

const string username\_pattern = "^[A-Za-z0-9]{3,15}$";

const string password\_pattern = "^[A-Za-z0-9]{8,}$";

const string email\_pattern = "^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";

const string mobile\_pattern = "^[0-9]{10}$";

// Validate each field

if (!validate\_regex(first\_name, name\_pattern)) {

cout << "Invalid first name" << endl;

}

if (!validate\_regex(last\_name, name\_pattern)) {

cout << "Invalid last name" << endl;

}

if (!validate\_regex(username, username\_pattern)) {

cout << "Invalid username" << endl;

}

if (!validate\_regex(password, password\_pattern)) {

cout << "Invalid password" << endl;

}

if (!validate\_regex(email, email\_pattern)) {

cout << "Invalid email" << endl;

}

if (!validate\_regex(mobile\_number, mobile\_pattern)) {

cout << "Invalid mobile number" << endl;

}

if (!validate\_city(city)) {

cout << "Invalid city" << endl;

}

// If all validations pass

if (validate\_regex(first\_name, name\_pattern) && validate\_regex(last\_name, name\_pattern) &&

validate\_regex(username, username\_pattern) && validate\_regex(password, password\_pattern) &&

validate\_regex(email, email\_pattern) && validate\_regex(mobile\_number, mobile\_pattern) &&

validate\_city(city)) {

cout << "Form data is valid!" << endl;

}

return 0;

}

// Function to validate a string against a regex pattern

bool validate\_regex(const string& str, const string& pattern) {

regex re(pattern);

return regex\_match(str, re);

}

// Function to validate the city

bool validate\_city(const string& city) {

// List of valid cities

vector<string> valid\_cities = {

"New York", "Los Angeles", "Chicago", "Houston", "Phoenix", "Philadelphia",

"San Antonio", "San Diego", "Dallas", "San Jose"

};

for (const auto& valid\_city : valid\_cities) {

if (city == valid\_city) {

return true;

}

}

return false;

}

**Question 4:**

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

