IV B. Tech - I Sem	Rust Programming	L	T	P	C
		0	0	4	2

COURSE OBJECTIVES

By the end of this course, students should be able to:

1. Define and explain key Rust programming concepts, including ownership, borrowing, and lifetimes.
2. Write, compile, and execute basic Rust programs, including applications with user-defined functions and control structures.
3. Create and manipulate variables of different data types in Rust. Use constants and mutable variables appropriately.
4. Define functions with parameters and return values. Organize code into modules and submodules.
5. Understand and apply concepts related to enums, pattern matching, and option/result types.
6. Create and implement traits to define common behavior for types.
7. Read and write files in Rust.

COURSE OUTCOMES

1. Understand the fundamental concepts of Rust, its strengths, and its role in modern software development.
2. Write Rust programs that use variables, constants, and control flow constructs.
3. Define and use functions, modules, and write Rustdoc documentation for code.
4. Understand ownership, borrowing, and lifetimes in Rust to write memory-safe programs.
5. Implement error handling strategies, create custom structs, and use enums and pattern matching effectively.
6. Write reusable code using traits and generics, and work with Rust collections and iterators.

Syllabus

Chapter 1

Introduction to Rust, The Basic Formatting, Printing Styles, Comments, Rust Variables, What Are Variables, Scope and Shadowing, Rust Data Types, What Are Data Types? Numeric Types: Integers and Floats, Boolean, Character and String, Arrays, Tuples, Constant Variables.

Programming Exercises

1. Write a program to display output using print!
2. Write a program to display Output following pattern using Placeholders

```
1
22
333
4444
```

55555

3. Write a program to do the following
    a. Declare a variable x and store value 1000 in it.
    b. Declare a variable y and store value “Programming” in it
    c. Print the values of x and y
    d. Change the value of x to 1100
    e. Print the values of x and y
4. Write a program to implement the Scope and Shadowing
5. Write a program to implement the following
    a. Implicit type declaration
    b. Explicit type declaration
6. Write Program to Declare an array, arr, of size 6 that has numbers divisible by 2 ranging from 0 to 10 and Print the value of arr.
7. Write a program to create and access a tuple.
8. Write a program to create an array of 10 elements and implement the following
    a. Create a of 2nd and 3rd element
    b. Omit the start index of the slice
    c. Omit the End Index of the Slice
    d. Omit both Start and End Index of the Slice
9. Write a program to create different types of constants print it in the output
10. Declaring String Object and converting String Literal to String Object

Chapter 2

Rust Operators, Introduction to Operators, Arithmetic Operators, Logical Operators, Comparison Operators, Bitwise Operators, Assignment and Compound Assignment Operators, Type Casting Operator, Borrowing and Dereferencing Operators, Precedence and Associativity.

Programming Exercises

1. Write a program to implement Type Casting Operator.
2. Write a program to implement Borrowing and Dereferencing Operators

Chapter 3

Rust if and if let Expressions, Introduction to Conditional Expression, If Expression, If Let Expression, Match Expression, Rust Loops, Introduction to Loops, Definite Loop - For Loop, Indefinite Loop - While and Loop, Break Statement, Continue Statement, Nested Loops, Loop Labels.

Programming Exercises

1. Write a program to check if a number is positive or negative
2. Write a program to determine if a number is even or odd
3. Write a program to make a calculator using Match Expression
4. Write a program to Match a pattern using If Let Expression.

5. Write a program to Print First 10 Natural Numbers using Loop
6. Write a program to Multiplication Table using Loop Labels
7. Write a program to Count Iterations of a Loop Until a Condition
8. Write a program to Print the following patterns
   &
   &&
   &&&
   &&&&
   &&&&&
9. Write a program to print the values in a collection using iter() method

Chapter 4

Introduction to Functions, Functions with Parameters, Pass by Value, Pass by Reference, returning a Value from a Function, Function with Multiple Return Values, Functions with Arrays as Arguments, Rust Strings, Introduction to Strings, Core Methods of String Objects, Iterating Over Strings, updating a String, Slicing a String, Functions and Strings.

Programming Exercises

1. Write a program to Find The Factorial using functions.
2. Write a function test_divisibility_by_3_4 which will check whether a given integer number is divisible by 3 or 4.
   a. If the number is divisible by both return 0
   b. If the number is divisible by 3 only return 1
   c. If the number is divisible by 4 only return 2
   d. If the number is not divisible by both, return -1
3. Write a program to demonstrate the Pass by Value and Pass by Reference
4. Write a function calculate_area_perimeter() that takes x and y(length and width of a rectangle) as a parameter to the function and returns a tuple (area, perimeter).
5. Write a function arr_square() that returns the Array of Squares
6. write a recursive function fibonacci that takes a positive integer number n as a parameter and returns the nth Fibonacci term in that range.
7. Write a program to Creating a String
8. Implement the string manipulation operations using Core Methods of String Objects
   a. str.capacity()
   b. str.contains("sub_str")
   c. str.replace(replace_from, replace_to)
   d. string.trim()
9. Write a program to tokenize and iterate over a string
10. Write a program to push a string into a string.

11. Write a program to find all words starting with a “c” in a string passed as a parameter. Concatenate them together and return the result.

Chapter 5

Rust Vectors, Introduction to Vectors, resizing a Vector, Iterating Over a Vector, slicing a Vector, Rust Structs, Introduction to Structs, Functions and Structs, Methods of Structs, Static Method of Structs, Tuple Structs, Enums.

Programming Exercises

1. Given a vector with an even number of elements, remove the last element from the input vector, and then the middle element. Then insert the sum of the remaining elements to the end of the resulting vector.
2. Write a program to Calculate Distance Between Two Points:
    a. A struct Point is given which has two items, x and y.
    b. The function test is given which has two instances of points initialised with some value of x and y.
    c. The task is to calculate the distance between the two points.
    d. The distance between two points is:

$$\sqrt{(x1-x2)^2+(y1-y2)^2}$$

    e. Return the value of distance
3. Write a program to Invoke a Static Method on Struct
4. Create a struct representing a person with fields such as name, age, and address. Implement methods to perform operations on the person.
5. Create a struct representing a geometric shape and implement methods to calculate its area and perimeter.
6. Write a program to Find If the Day Is a Weekend
    a. An enum Days has been provided to you. It has all the days of the week.
    b. A method is_weekend() is incomplete.
    c. The task is to complete the method
    d. If the day is a weekend it returns 1
    e. If the day is a weekday it returns 0
7. Define an enum representing different types of geometric shapes (e.g., circle, square, triangle). Implement a function that calculates the area of a shape based on its variant.
8. Define an enum representing different types of vehicles and use pattern matching to print their respective characteristics.

Chapter 6

Traits and Modules, Ownership and Borrowing, File I/O and Error Handling, Testing, Async, Unsafe, Macros, Threads.

Programming Exercises

1. Write a program that implements Traits
    a. Define a trait named `Animal`
    b. Define a method `make_sound` that needs to be implemented by types implementing the
    c. Implement the `Animal` trait for the `Dog` struct which prints “Woof!”
    d. Implement the `Animal` trait for the `Cat` struct which prints “Meow!”
    e. Define A function ‘animal_sounds’ that takes any type implementing the `Animal` trait
2. Write a program to Find the Area of a Triangle using modules
3. Write a program that demonstrates ownership transfer by creating variables and passing them between functions.
4. Experiment with different scenarios to understand the Rust compiler's behaviour in terms of ownership and borrowing.
5. Write a program to read data from a file and write the results back to another file using Rust's file I/O capabilities.
6. Rust program that demonstrate different error handling techniques
7. Rust program to implement Recoverable Errors with Result
8. Write unit tests for a Rust module, covering various functionalities and edge cases.
9. Develop a program that makes asynchronous HTTP requests using a Rust async runtime and processes the responses.
10. Create a program that uses unsafe Rust code to interface with a C library or perform low-level memory operations.
11. Define custom macros in Rust to automate repetitive code generation or introduce domain-specific language constructs.
12. Implement a program that uses threads to perform parallel computations on a large dataset.