Principles of programming languages

Sub Code: 20CYS312 **Name: Chitra Harini**

Roll No:CH.EN.U4CYS22010 **Date: 21-02-2025**

GITHUB LINK: https://github.com/Harini-chitra/Haskell/tree/main/Haskell_Lab7

Lab 7

Task 1: Data Types and Variables.

Objective: Declare variables of the following types: integer, floating-point, boolean, and character. Print the value of each variable.

Rust Code:

```
fn main() {
    let int_var: i32 = 22010;
    let float_var: f64 = 22.33;
    let bool_var: bool = false;
    let char_var: char = 'H';

    println!("Integer: {}", int_var);
    println!("Floating-point: {}", float_var);
    println!("Boolean: {}", bool_var);
    println!("Character: {}", char_var);
}
```

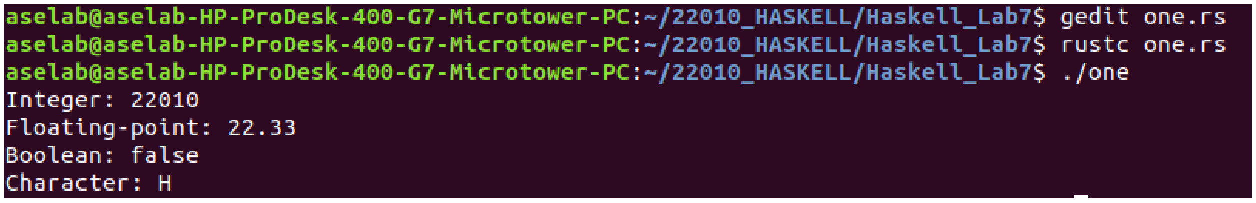
Explanation of code:

- Declares variables of integer (i32), floating-point (f64), boolean (bool), and character (char) types.
- Prints the values using println!.

I/O Examples:

```
Integer: 22010
Floating-point: 22.33
Boolean: false
Character: H
```

Output Screenshot:



```
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ gedit one.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ rustc one.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ ./one
Integer: 22010
Floating-point: 22.33
Boolean: false
Character: H
```

Task 2: Simple Arithmetic Operations

Objective: Declare two integer variables and perform the following operations:

a. Addition
b. Subtraction
c. Multiplication
d. Division
e. Modulo

Print the result of each operation.

Rust Code:

```
fn main() {
   let a: i32 = 10;
   let b: i32 = 5;

   println!("Addition: {}", a + b);
   println!("Subtraction: {}", a - b);
   println!("Multiplication: {}", a * b);
   println!("Division: {}", a / b);
   println!("Modulo: {}", a % b);
}
```

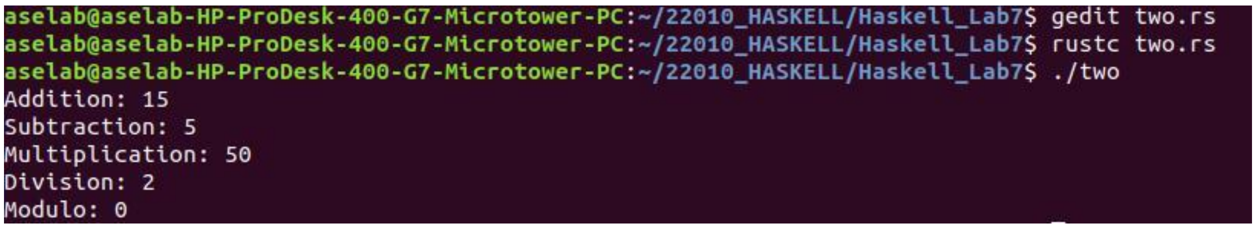
Explanation of code:

- Declares two integer variables a and b.
- Performs and prints results of addition, subtraction, multiplication, division, and modulo.

I/O Examples:

```
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2
Modulo: 0
```

Output Screenshot:



```
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ gedit two.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ rustc two.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ ./two
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2
Modulo: 0
```

Task 3: If-Else Decision Making

Objective: Write a program that:

a. Takes a number as input.
b. Checks whether the number is positive, negative, or zero using an if-else statement.
c. Print a message based on the result.

Rust Code:

```
fn check_number(n: i32) {
   if n > 0 {
      println!("Positive");
   } else if n < 0 {
      println!("Negative");
   } else {
      println!("Zero");
   }
}

fn main() {
   check_number(-90);
   check_number(22);
   check_number(0);
}
```

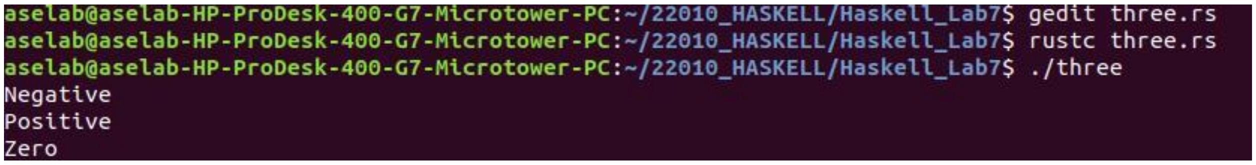
Explanation of code:

- The function check_number checks if n is positive, negative, or zero.
- The main function calls check_number with different values.

I/O Examples:

```
Negative
Positive
Zero
```

Output Screenshot:



```
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ gedit three.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ rustc three.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ ./three
Negative
Positive
Zero
```

Task 4: Checking for Even or Odd

Objective: Write a program that:

a. Takes an integer as input.
b. Uses an if-else statement to check if the number is even or odd.
c. Print "Even" if the number is even and "Odd" if the number is odd.

Rust Code:

```
fn check_even_odd(n: i32) {
if n % 2 == 0 { println!("Even");
 } else {
println!("Odd"); }
}
fn main() {
check_even_odd(2);
check_even_odd(9);
check_even_odd(0);
}
```

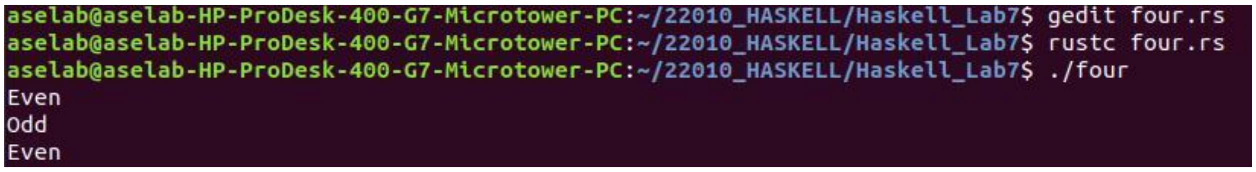
Explanation of code:

- Uses the modulus operator % to check if a number is even or odd.
- Calls check_even_odd with different values in main.

I/O Examples:

```
Even
Odd
Even
```

Output Screenshot:



```
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ gedit four.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ rustc four.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ ./four
Even
Odd
Even
```

Task 5: Using a Loop to Print Numbers

Objective: Write a program that uses a for loop to print the even numbers from the range 1 to 20.

Rust Code:

```
fn main() {
    for num in 1..=20 {
        if num % 2 == 0 {
            println!("{}", num);
        }
    }
}
```

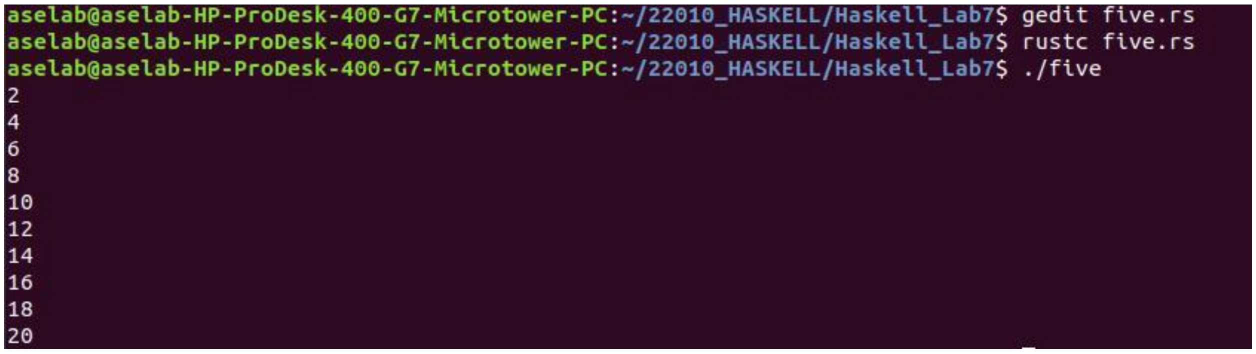
Explanation of code:

- Loops from 1 to 20 using for num in 1..=20.
- Prints numbers if divisible by 2.

I/O Examples:

```
2
4
6
8
10
12
14
16
18
20
```

Output Screenshot:



```
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ gedit five.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ rustc five.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ ./five
2
4
6
8
10
12
14
16
18
20
```

Task 6: While Loop Example

Objective: Write a program that uses a while loop to print odd numbers from the range 1 to 20.

Rust Code:

```
fn main() {
    let mut num = 1;
    while num <= 20 {
        if num % 2 != 0 {
            println!("{}", num);
        }
        num += 1;
    }
}
```

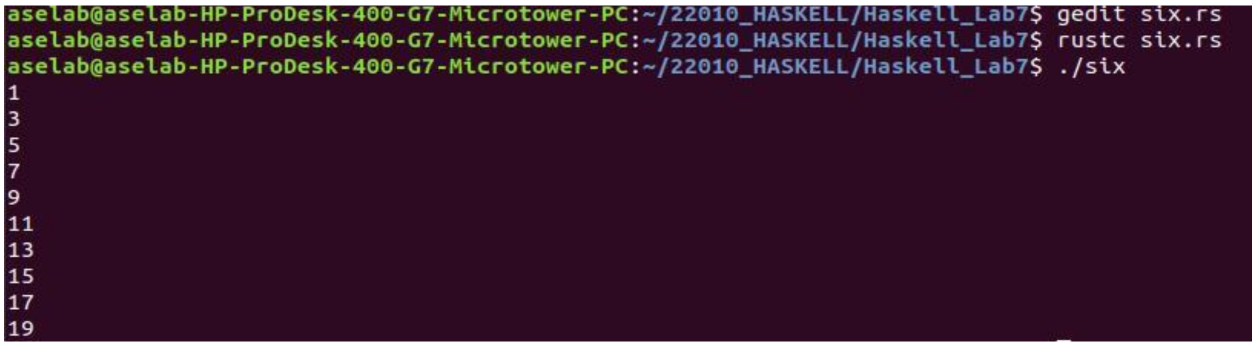
Explanation of code:

- Uses a while loop to iterate from 1 to 20.
- Prints numbers if they are odd (num % 2 != 0).

I/O Examples:

```
1
3
5
7
9
11
13
15
17
19
```

Output Screenshot:



```
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ gedit six.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ rustc six.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ ./six
1
3
5
7
9
11
13
15
17
19
```

Task 7: Using a For Loop with a Range

Objective: Write a program that uses a for loop to print the numbers from 10 to 1 in reverse order (10, 9, 8, ..., 1).

Rust Code:

```
fn main() {
    for num in (1..=10).rev() {
        println!("{}", num);
    }
}
```

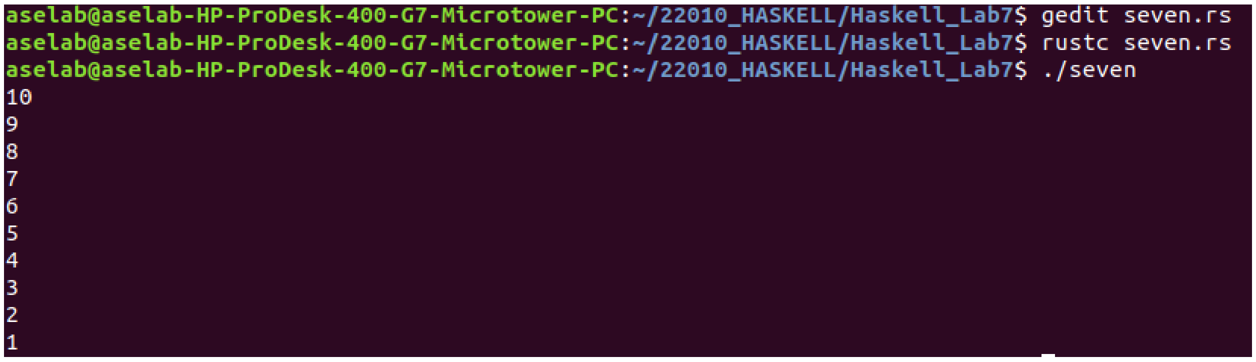
Explanation of code:

- Uses (1..=10).rev() to loop from 10 to 1 in reverse order.
- Prints each number.

I/O Examples:

```
10
9
8
7
6
5
4
3
2
1
```

Output Screenshot:



```
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ gedit seven.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ rustc seven.rs
aselab@aselab-HP-ProDesk-400-G7-Microtower-PC:~/22010_HASKELL/Haskell_Lab7$ ./seven
10
9
8
7
6
5
4
3
2
1
```