

Department of Cyber Security
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Chennai Campus
Principals of Programming Languages

Subject Code: 20CYS312

Date:2024/11/29

Name: Sushant Yadav

Roll Number:CH.EN.U4CYS22067

Lab 1: Introduction to Programming Paradigms (Simple Programs in Haskell)

Part 1: Haskell Exercises (45 minutes)

1. Basic Arithmetic:

Objective: Get familiar with GHCi and basic arithmetic operations.

Exercise 1: Open GHCi and perform basic arithmetic operations:

Haskell Code

$3 + 5$

$10 * 4$

$6 / 2$

Output:

```
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ ghci  
GHCi, version 9.4.8: https://www.haskell.org/ghc/ :? for help  
ghci> 5+3  
8  
ghci> 10*4  
40  
ghci> 6/2  
3.0  
ghci>
```

Exercise 2: Define a function to calculate the square of a number:

Open Terminal.

Create a file with the .hs extension using nano (or your preferred text editor).
Or(nano.file name .hs).

Write the following code inside the square.hs file & Press Ctrl + X to exit nano.

Press Y to confirm saving the file, then press Enter to confirm the filename (square.hs).

Compile the program by running the following command:

```
ghc -o square square.hs  
./square
```

Haskell Code

```
square :: Int -> Int
```

```
square x = x * x
```

```
main :: IO ()
```

```
main = print (square 5)
```

Output:

```
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ nano square.hs  
  
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ ghc -o square square.hs  
[1 of 2] Compiling Main                ( square.hs, square.o )  
[2 of 2] Linking square.exe  
  
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ ./square  
25
```

2. Defining and Using Lists:

Objective: Understand basic data structures like lists in Haskell.

Exercise 3: Create a list of numbers and compute the sum of the list:

Haskell Code

```
sumList :: [Int] -> Int
```

```
sumList [] = 0  
sumList (x:xs) = x + sumList xs
```

Test with: `sumList [1, 2, 3, 4, 5]`

Output:

```
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ nano list.hs  
  
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ ghc -o List List.hs  
[1 of 2] Compiling Main (List.hs, List.o)  
[2 of 2] Linking List.exe  
  
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ ./list  
15  
  
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~  
$ |
```

3. Pattern Matching with Lists:

Objective: Learn how pattern matching works in Haskell.

Exercise 4: Write a function to check if a list is empty:

Haskell Code

```
isEmpty :: [a] -> Bool
```

```
isEmpty [] = True
```

```
isEmpty _ = False
```

Test with: `isEmpty [1, 2, 3]` and `isEmpty []`.

Output:

```

Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
$ nano empty.hs

Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
$ ghc -o empty empty.hs
[1 of 2] Compiling Main             ( empty.hs, empty.o )
[2 of 2] Linking empty.exe

Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
$ ./empty
False

Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
$ nano empty.hs

Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
$ ghc -o empty empty.hs
[1 of 2] Compiling Main             ( empty.hs, empty.o ) [Source file changed]
[2 of 2] Linking empty.exe [Objects changed]

Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
$ ./empty
True

```

4. Simple IO Operations:

Objective: Understand basic input and output in Haskell.

Exercise 5: Write a program that asks the user for their name and prints a greeting:

haskell

Copy code

```
main :: IO ()
```

```
main = do putStrLn "What is your name?"
```

```
name <- getLine
```

```
putStrLn ("Hello, " ++ name)
```

Output:

```
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
```

```
$ nano haskell.hs
```

```
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
```

```
$ ghc -o haskell haskell.hs
```

```
[1 of 2] Compiling Main
```

```
( haskell.hs, haskell.o )
```

```
[2 of 2] Linking haskell.exe
```

```
Lenovo@LAPTOP-BQHD45S5 MINGW64 ~
```

```
$ ./haskell
```

```
What is your name?
```

```
Sushant yadav
```

```
Hello, Sushant yadav
```