20CYS312 -PRINCIPLE OF PROGRAMMING LANGUAGES

Date: 29-11-2024

Name: Naren S

Roll no.: CH.EN.U4CYS22034

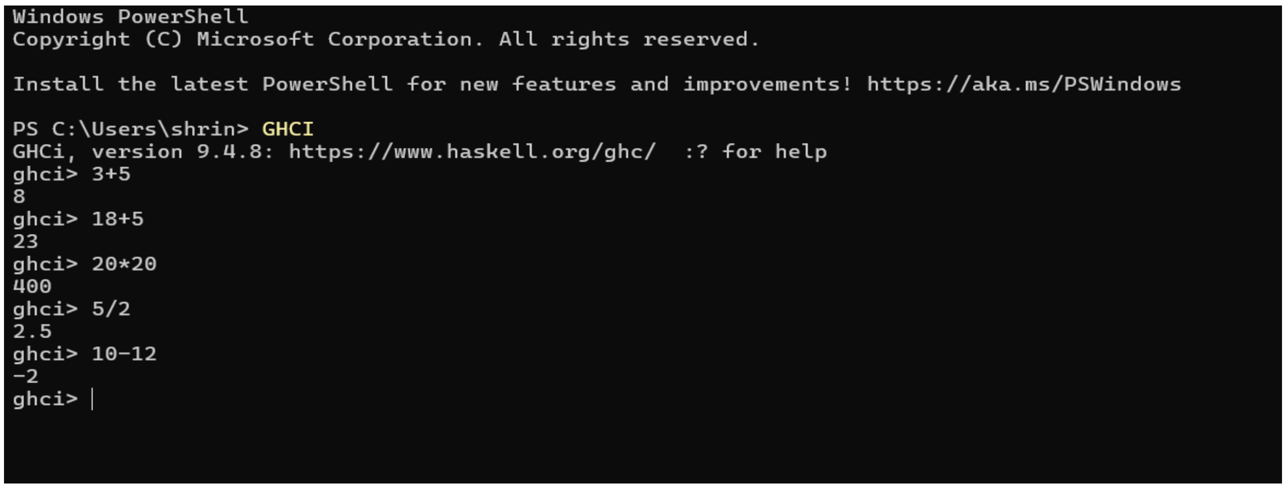
LAB-1

Part 1: Haskell Exercises

1. Basic Arithmetic:

Objective: Get familiar with ghci and basic arithmetic operations.

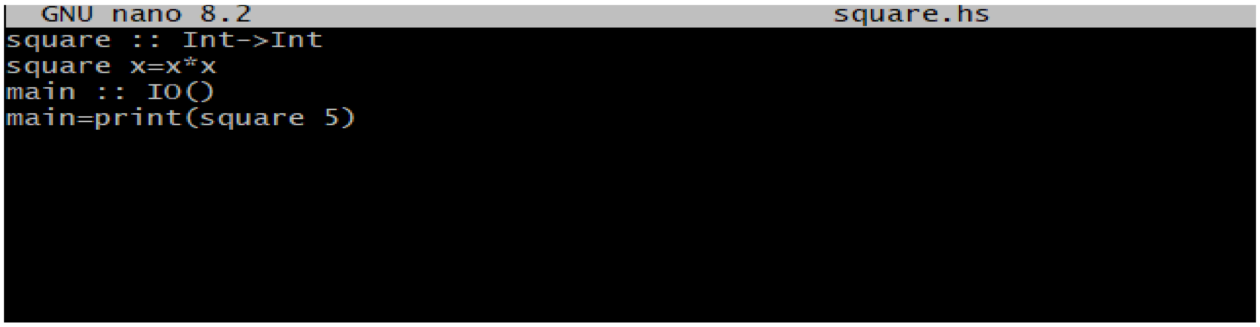
Exercise 1: Open ghci and perform basic arithmetic operations:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\shrin> GHCI
GHCi, version 9.4.8: https://www.haskell.org/ghc/  :? for help
ghci> 3+5
8
ghci> 18+5
23
ghci> 20*20
400
ghci> 5/2
2.5
ghci> 10-12
-2
ghci> |
```

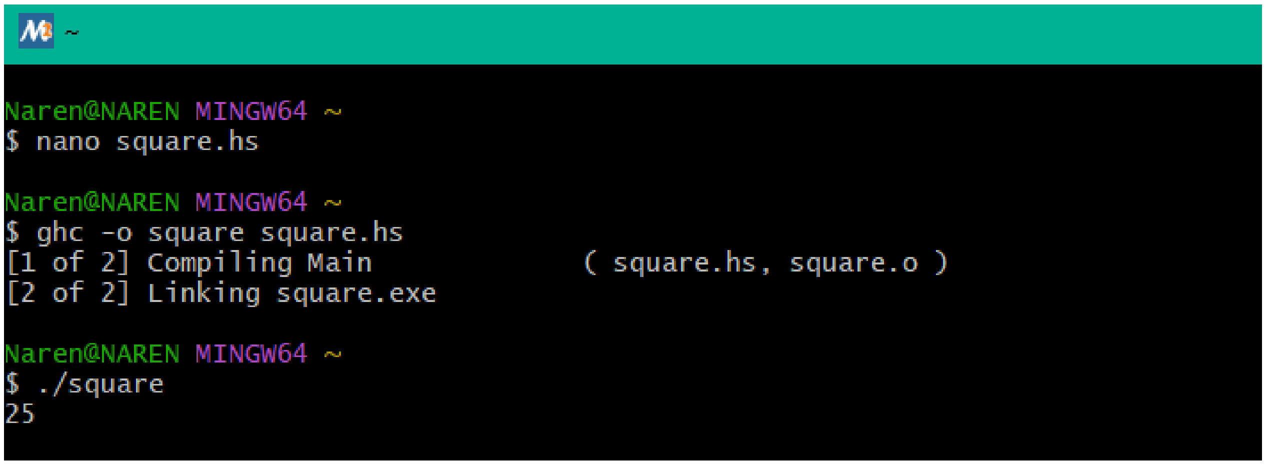
Exercise 2: Define a function to calculate the square of a number:

Code:



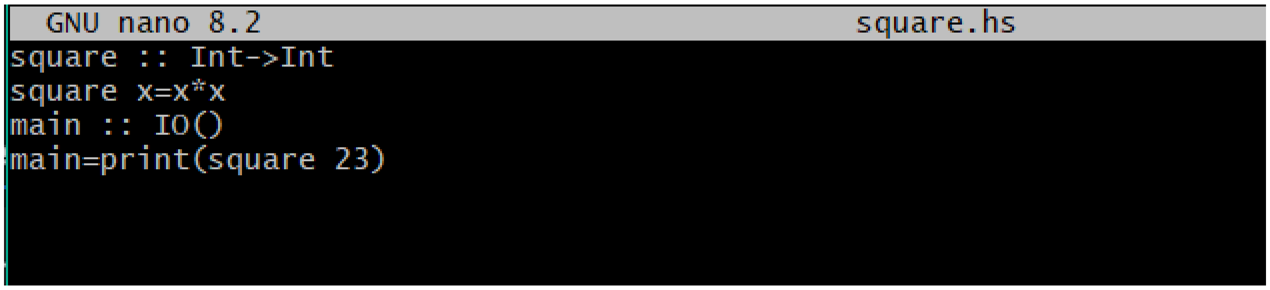
```
  GNU nano 8.2                                     square.hs
square :: Int->Int
square x=x*x
main :: IO()
main=print(square 5)
```

Output:



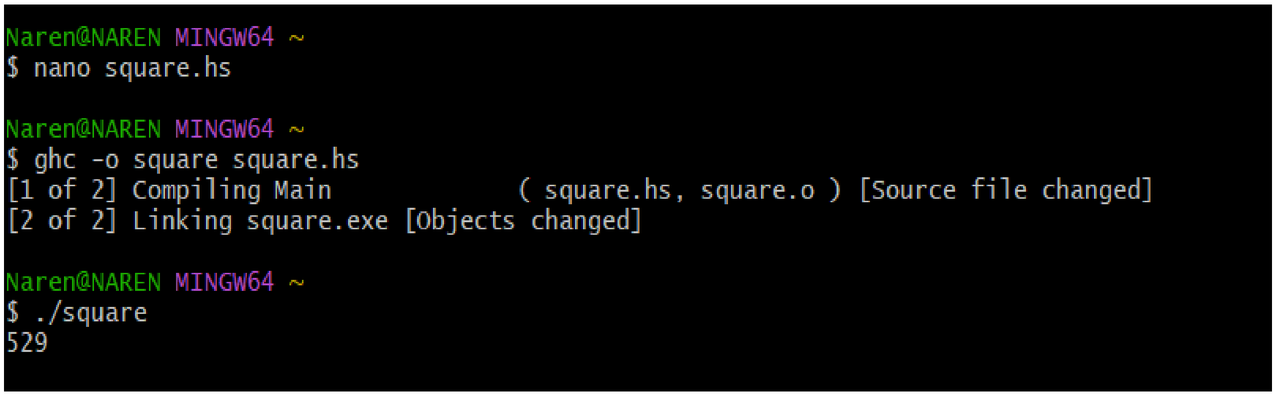
```
~

Naren@NAREN MINGW64 ~
$ nano square.hs

Naren@NAREN MINGW64 ~
$ ghc -o square square.hs
[1 of 2] Compiling Main             ( square.hs, square.o )
[2 of 2] Linking square.exe

Naren@NAREN MINGW64 ~
$ ./square
25
```

Code:



```
 GNU nano 8.2                                    square.hs
square :: Int->Int
square x=x*x
main :: IO()
main=print(square 23)
```

Output:



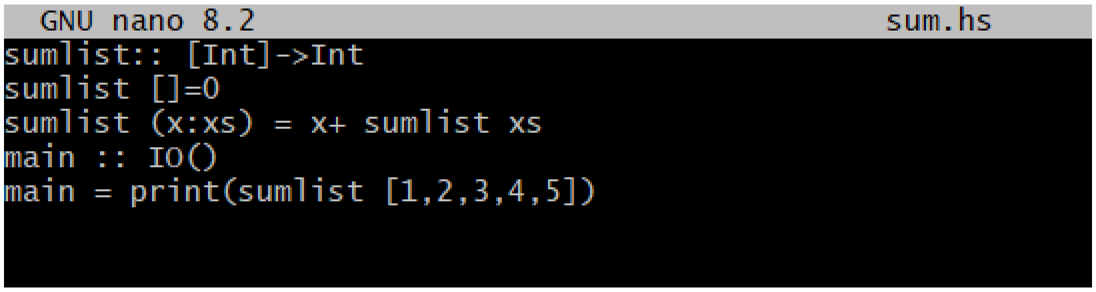
```
Naren@NAREN MINGW64 ~
$ nano square.hs

Naren@NAREN MINGW64 ~
$ ghc -o square square.hs
[1 of 2] Compiling Main             ( square.hs, square.o ) [Source file changed]
[2 of 2] Linking square.exe [Objects changed]

Naren@NAREN MINGW64 ~
$ ./square
529
```

2. Defining and Using Lists:

Objective: Understand basic data structures like lists in Haskell.

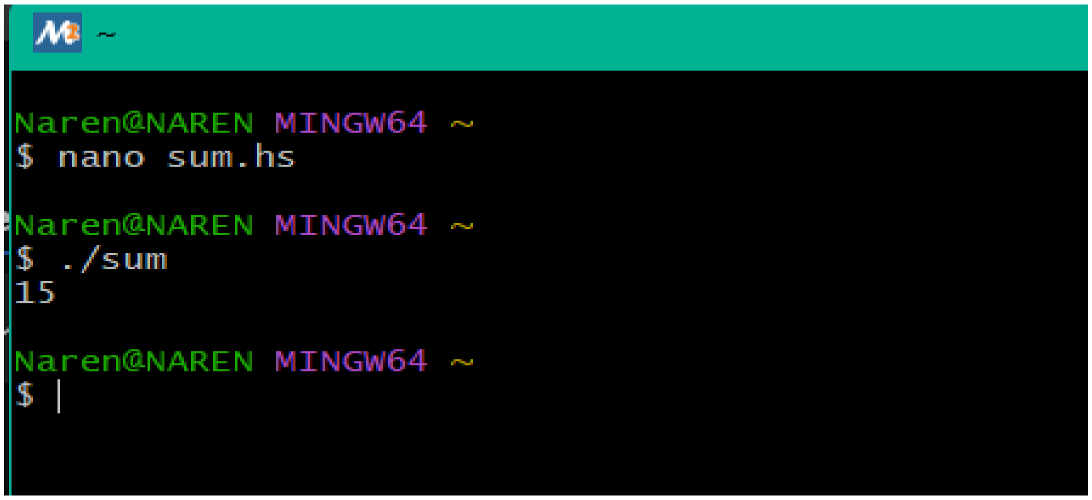
Exercise 3: Create a list of numbers and compute the sum of the list:

Code:



```
  GNU nano 8.2                                    sum.hs
sumlist:: [Int]->Int
sumlist []=0
sumlist (x:xs) = x+ sumlist xs
main :: IO()
main = print(sumlist [1,2,3,4,5])
```

Output:



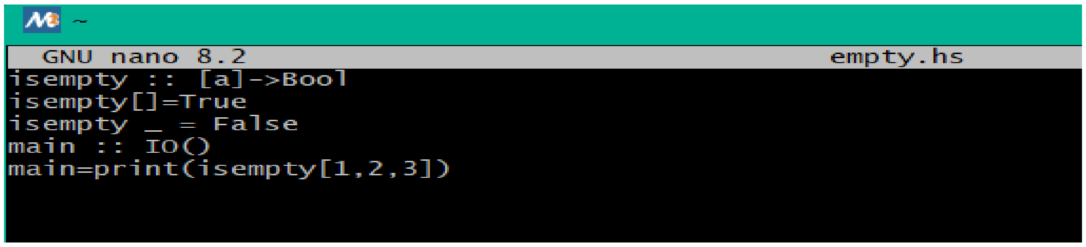
```
Naren@NAREN MINGW64 ~
$ nano sum.hs

Naren@NAREN MINGW64 ~
$ ./sum
15

Naren@NAREN MINGW64 ~
$ |
```

3. Pattern Matching with Lists:

Objective: Learn how pattern matching works in Haskell.

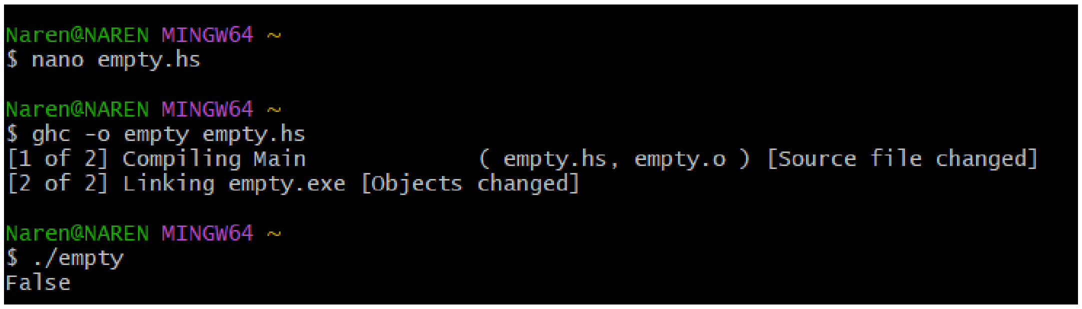
Exercise 4: Write a function to check if a list is empty:

Code for (isempty[1,2,3]):



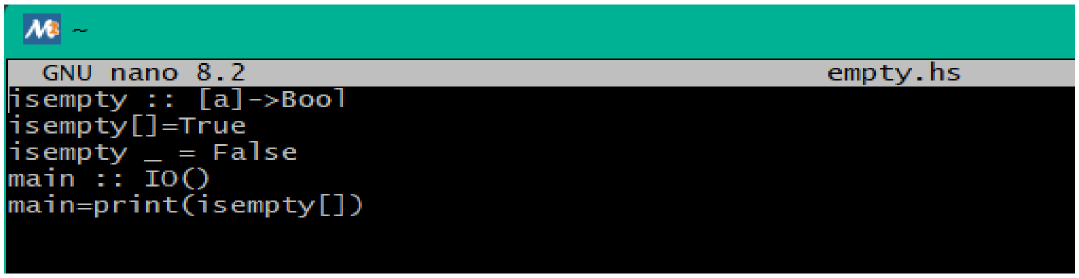
```
  GNU nano 8.2                                  empty.hs
isempty :: [a]->Bool
isempty[]=True
isempty _ = False
main :: IO()
main=print(isempty[1,2,3])
```

Output:



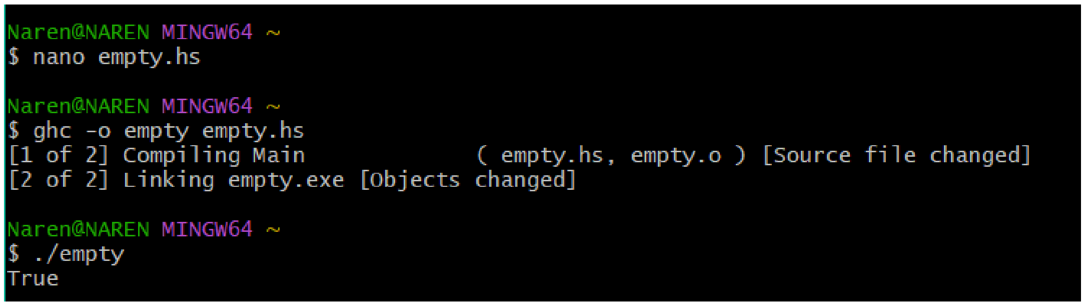
```
Naren@NAREN MINGW64 ~
$ nano empty.hs

Naren@NAREN 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]

Naren@NAREN MINGW64 ~
$ ./empty
False
```

Code for (isempty[]):



```
  GNU nano 8.2                                       empty.hs
isempty :: [a]->Bool
isempty[]=True
isempty _ = False
main :: IO()
main=print(isempty[])
```

Output:



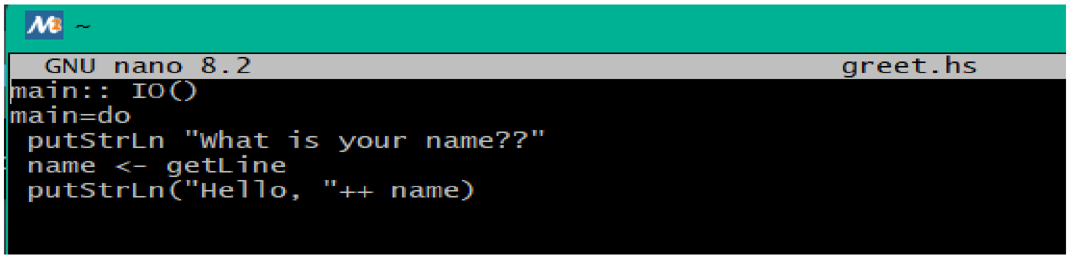
```
Naren@NAREN MINGW64 ~
$ nano empty.hs

Naren@NAREN 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]

Naren@NAREN MINGW64 ~
$ ./empty
True
```

4) Simple IO Operations:

Objective: Understand basic input and output in Haskell.

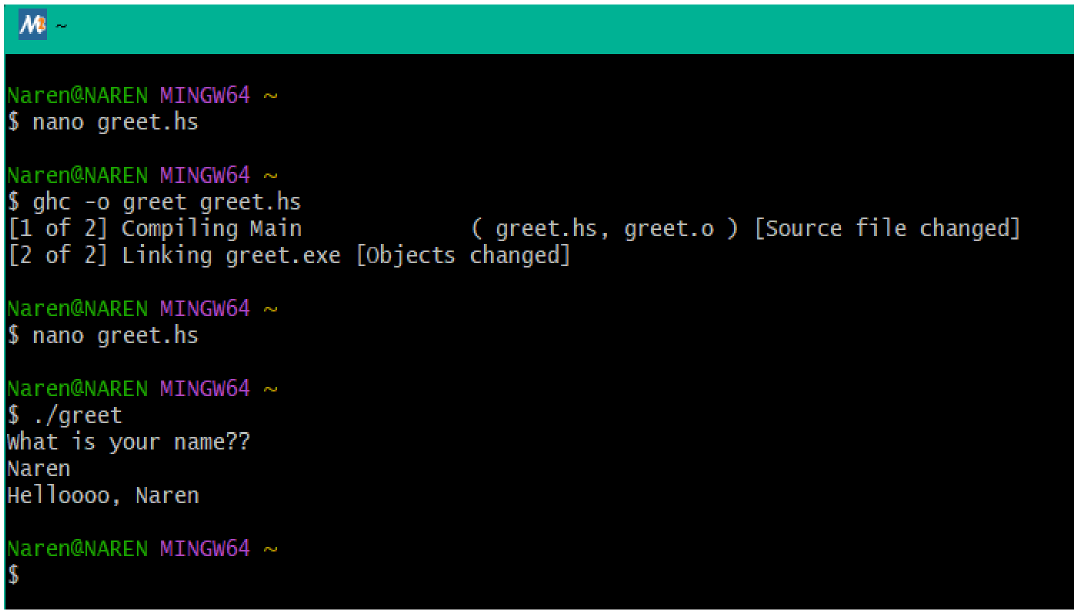
Exercise 5: Write a program that asks the user for their name and prints a greeting:

Code:



```
  GNU nano 8.2                                       greet.hs
main:: IO()
main=do
 putStrLn "What is your name??"
 name <- getLine
 putStrLn("Hello, "++ name)
```

Output:



```
Naren@NAREN MINGW64 ~
$ nano greet.hs

Naren@NAREN MINGW64 ~
$ ghc -o greet greet.hs
[1 of 2] Compiling Main             ( greet.hs, greet.o ) [Source file changed]
[2 of 2] Linking greet.exe [Objects changed]

Naren@NAREN MINGW64 ~
$ nano greet.hs

Naren@NAREN MINGW64 ~
$ ./greet
What is your name??
Naren
Helloooo, Naren

Naren@NAREN MINGW64 ~
$
```