FUNCTIONAL PROGRAMMING

Haskell

MODULE

- Haskell is a functional language and everything is denoted as an expression, hence a **Module** can be called as a **collection of similar or related types of functions**.

- You can **import** a function from one module into another module.

- All the "**import**" **statements should come first** before you start defining other functions.

LIST MODULE

```
import Data.List

main = do
  putStrLn("Different methods of List Module")
  print(intersperse '.' „Haskell.com")
  print(intercalate " " ["Lets","Start","with","Haskell"])
  print(splitAt 7 "HaskellTutorial")
  print (sort [8,5,3,2,1,6,4,2])
```

Output:

```
Different methods of List Module
"H.a.s.k.e.l.l...c.o.m"
"Lets Start with Haskell"
("Haskell","Tutorial")
[1,2,2,3,4,5,6,8]
```

CHAR MODULE

The **Char** module has plenty of predefined functions to work with the Character type. Take a look at the following code block −

```
import Data.Char

main = do
  putStrLn("Different methods of Char Module")
  print(toUpper 'a')
  print(words "Let us study tonight")
  print(toLower 'A')
```

MAP MODULE

Map is an unsorted value-added pair type data type. It is a widely used module with many useful functions. The following example shows how you can use a predefined function available in the Map module.

```
import Data.Map (Map)
import qualified Data.Map as Map  --required for GHCI

myMap :: Integer -> Map Integer [Integer]
myMap n = Map.fromList (map makePair [1..n])
   where makePair x = (x, [x])

main = print(myMap 3)
```

FILES AND STREAMS

Let us create a file and name it "xyz.txt". Next, enter the following lines in this text file: "Welcome to Haskell".

Next, we will write the following code which will display the contents of this file on the console. Here, we are using the function readFile() which reads a file until it finds an EOF character.

```
main = do
  let file = "xyz.txt"
  contents <- readFile file
  putStrLn contents
```

FILES AND STREAMS (CONTINUED..)

Write in a file:

```
main = do
 let file = "abc.txt"
 writeFile file "I am just experimenting here."
 readFile file
```

TYPECLASS

- Typeclasses are among the most powerful features in Haskell.
- They allow us to define generic interfaces that provide a common feature set over a wide variety of types.
- Typeclasses are at the heart of some basic language features such as equality testing and numeric operators.

```
data Color = Red | Green | Blue
colorEq :: Color -> Color -> Bool
colorEq Red Red = True
colorEq Green Green = True
colorEq Blue Blue = True
colorEq _ _ = False
```

RECORD SYNTAX

- We've been tasked with creating a data type that describes a person. The info that we want to store about that person is: first name, last name, age, height, phone number, and favorite ice-cream flavor.

data Person = Person String String Int Float String String deriving (Show)

Output:

```
ghci> let guy = Person "Buddy" "Finklestein" 43 184.2 "526-2928" "Chocolate"
ghci> guy
Person "Buddy" "Finklestein" 43 184.2 "526-2928" "Chocolate"
```

```
data Car = Car {company :: String, model :: String, year :: Int} deriving (Show)
```

Output:

```
ghci> Car {company="Ford", model="Mustang", year=1967}
Car {company = "Ford", model = "Mustang", year = 1967}
```

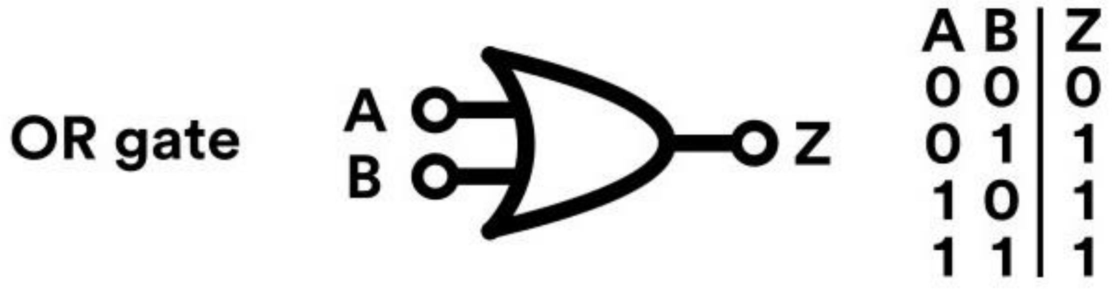
PRACTICE

- Write a program based on **Map Module** in haskell, which produces the following output:

  [(1,[4]) ,(2,[8]), (3,[12]), (4,[16])]

- Write a program in haskell to find a number is divisible by 5 or not and will **write in a file** either "divisible by 5" when divisible by 5 OR "not divisible by 5" when it is not divided by 5.

- Write a program in haskell which produces output of **OR gate** using **TYPECLASS.**



A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

- Write a program in haskell to add a **record** for a book with the following information:

Book name, Author name, ISBN number, Year of publishing and Version Number.

ASSIGNMENT (DEADLINE: 26TH NOVEMBER)

- Write a program in haskell that determines a String is palindrome or not.

Examples of some palindroms: Anna, Civic, Kayak, Level, Madam etc.

- Write a haskell program to remove the duplicates from a given list of integers.

Expected Output:

Original list:

1 1 2 3 4 4 5 6 6 6

After removing duplicates from the above list:

1 2 3 4 5 6