Experiment No: 01

Aim: Tutorial on Haskell Programming Environment

Lab Objective: Design and implement declarative programs in functional and logic programming languages.

Lab Outcomes: Design and Develop solution based on declarative programming paradigm using functional and logic programming (LO2)

Requirements: Any Text Editor and Glasgow Haskell Compiler 8.0+ Version

Theory:

Programming Paradigm refer to style of building the structure and elements of computer programs. Haskell Programming language belongs to Functional Paradigms which is one of the popular paradigms under Declarative Programming. In functional programming computation is treated as evaluation of mathematical functions and avoids changing state of computation by mutating the data.

Some important advantages of using functional programming include:

- It is based on mathematical functions.
- Easier to identify inputs and outputs to a computational task.
- Easier to demonstrate and prove that we have correct program to solve the computational task.
- Easier to test programs that are otherwise difficult to prove.

Few limitations of Functional Programming include:

- It is difficult to understand, writing a functional code for beginner's.
- Hard to maintain as many objects evolve during the coding.
- Re-use is very complicated and needs constant refactoring.
- Objects may not represent the problem correctly.

Haskell is widely used purely functional programming language. Few points that make this language so special over other conventional programming languages such as Java, C, C++, PHP, etc are:

- **Functional Language** − In conventional programming language, we instruct the compiler a series of tasks but in Haskell we will tell our computer "what it is?" in terms of a function definition.
- **Laziness** − Haskell is a lazy language. By lazy, we mean that Haskell won't evaluate any expression without any reason. When the evaluation engine finds that an expression needs to be evaluated, then it creates a data structure to collect all the required information for that specific evaluation and a pointer to that data structure. The evaluation engine will start working only when it is required to evaluate that specific expression.
- **Modularity** − A Haskell application is nothing but a series of functions. We can say that a Haskell application is a collection of numerous small Haskell applications. This makes Haskell application produce modular solution.
- **Statically Typed** − In conventional programming language, we need to

define a series of variables along with their type. In contrast, Haskell has a well defined type inference system. By the term, type inference language, we mean the Haskell compiler is intelligent enough to figure out the type of the variable declared, hence we need not explicitly mention the type of the variable used. The type inference system determines type if all programming objects at compiler time and hence Haskell is statically typed programming language.

- **Maintainability** − Since Haskell applications are modular, it is very easy and cost-effective to maintain them.
- Functional programs are more concurrent and they follow parallelism in execution to provide more accurate and better performance. Haskell is no exception; it has been developed in a way to handle multi threading effectively.

Performance: [Note: While writing the write up student need to change the wording such that it coveys that students have done all following steps. Also where ever output is generated the output must be written by the student]

Students need to perform following steps to complete this Experiment:

1. Download and install Glasgow Haskell Compiler
2. Use GHCI tool that comes with Compiler installation to understand various operator types in Haskell
   a) Arithmetic Operators.
   b) Logical Operators
   c) Relational Operators
   d) Identify types of different programming objects in Haskell

Part 1. Installation of Glasgow Haskell Compiler on Ubuntu Desktop

1. Open Terminal Application
2. Type “sudo apt-get install ghc” and hit enter key. This command requires you to provide administrative password
3. This command installs ghc and ghci tools. ghc being the glasgow haskell compiler tool which can be used to compile Haskell file (.hs) into executable binary code and ghci is tool using which we can execute single Haskell statement and get output of the same in REPL (Read Evaluate Print Loop) manner.

Part 2. Using ghci tool.

1. To start interactive session type ghci on command line. Once the session start the prompt string changes to “**Prelude>**”
2. Once the prompt changes one can start using the interactive haskell environment to type and execute Haskell statement/s. First we will learn how to configure this environment.
3. There are many configuration settings that can be tweaked to start configuration mode one has to type ‘**:**’ and enter key word to mention configuration. e.g. to set up a text editor inside ghci one need to use keyword “**set editor**” and type the editor name. Following command sets the editor **nano** to be used as editor within ghci.

Prelude>:set editor nano <enter>

4. To open a haskell file for editing one can use following command.
   Prelude>:edit file.hs <enter>
   This command opens/creates file.hs file for editing in nano editor so that one can write haskell code. To close the editor one uses CTRL+X types correct file name, saves the file and exits.
5. To execute terminal commands in side ghci one has to use '**:!**' and the shell command.
   Prelude>:! ls <enter>
   This will list out all files and folders in present working directory.
6. One needs to type ':quit <enter>' to exit from ghci session.
7. Arithmetic Operators in Haskell:
   To understand various arithmetic operators in Haskell type following commands in ghci and note the output.
   a. 2*3 b. 2 *(-3) c. 2-3 d. ¾ e. 2^6 f. it g.
8. Logical Operators in Haskell: Haskell supports Boolean data type with two values **True** and **False** to mean logically True and False respectively. Type following commands in ghci and note the output.
   a. False b. True c. True && False d. True && False e. not False f. not (True && False)
9. To know about various relational operators in Haskell type following statements and execute in ghci
   a. 4 == 6 b. 5.0 == 5.0 c. 2 /= 4 d. 2 /= 2 e. 2 < (-6)
   f. 12 > 3 g. 'g' =='g' h. "Fgh" == "Fgh"
10. Haskell offers very rich type inference system which is statically typed and organizes various similar type data into categories called as **Type Class.** To know type of any constant one can use **:type <object name> <enter>** command in ghci. Note output of following commands
    a. :type False b. :type 4 c. :type 56.0 d. :type (-8) e. :type 'H'
    f. :type "Hello" g. type: 67.7898

Exersize: Answer following question:

1. Explain what are basic data types in Haskell?
2. Explain what is type signature ? What are components of a type signature?
3. Differentiate between Imperative and Functional Programming.

Conclusion: Thus we have learned Glasgow Haskell Compiler and use GHCI tool to execute Haskell statements interactively in REPL shell.

Reference:

1. Glasgow Haskell Project Home Page. https://www.haskell.org/
2. Learn You a Haskell for Great Good ! A Beginner's Guide http://learnyouahaskell.com/
3. Michael L Scott, 'Programming Language Pragmatics', 3rd Edition, Elsevier Publication.