June 23 - 25, 2022 Hackathon

Introduction

The goal of this initiative is to define the necessary content to bring young programmers and others to take an interest in Haskell. At the end of this Hackathon session, participants should be able to understand and apply (in a practical way) the various concepts introduced. Overall, it is about preparing participants to learn how to implement solutions to concrete life problems using the functional programming paradigm.

Participants Profile

This program is intended for anyone interested in software programming using the functional programming paradigm. There are no specific prerequisites to participate in this program, however it is recommended to have at least some programming experience.

Participants Readiness

To be more productive during this Hackathon, it is advised to prepare yourself by doing the following:

1. Do some preliminary research on Computer Programming, Sets and Functions in Maths and their relation to the Haskell programming language
2. Set up your working environment (Install Haskell and a text editor of your choice): https://www.haskell.org/downloads/, (optional)
3. Familiarise yourself with **repl it**, which can be found here: https://replit.com

Schedule

The hackathon will take place over the course of three days. The first two day will be reserved for theoretical lectures and practical exercises. The courses will be held live on Zoom with break rooms in English and French. The sessions will also be available immediately after on DITC and Wada's YouTube channels for those who cannot participate live. The third day will be devoted to a challenge session. The challenge consists of two parts: individual and group challenges. The individual challenge will be a MCQ (Multiple Choice Question) type test and the group challenge will be working in teams to implement a solution to a problem using Maths notions and / or Haskell (focusing on the concepts acquired on the first two days).

Prize distribution Total 1000 Ada

Each candidate will receive a prize for participation (proportional to the number of days attended and the number of participants).
For the individual and group challenge, prizes will be awarded in proportion to the scores obtained.

*The bonuses will be in Ada (CryptoCurrency) and will be transferred to candidates' electronic wallets a few days later following the event.

Course / Lecture Content

Given that our main goal is to spark interest in the use of Haskell, we have outlined the following content: (We are assuming that participants already have a working development environment).

What is Programming

1. Definition of Programming
2. Instructions
3. Programing language
4. Source code
5. Compilation
6. Execution
7. Programing Paradigms
    - Imperative
        i. Structured
        ii. Procedural
    - Object-oriented programming
        i. Prototype oriented
        ii. Class oriented
        iii. Component oriented
    - Declarative
        i. Descriptif
        ii. Logical
        iii. Functional

Maths Prerequisites

1. Elements of Set Theory
    - Definition of a set
    - Empty set
    - Sets Cardinality
    - Set Inclusion
    - Sets Equality
    - Subsets
    - Set Operations
        i. Intersection
        ii. Union
    - Set Properties
        i. Idempotence
        ii. Commutativity
        iii. Associativity
    - Difference
    - Symmetric Difference
    - Complementary
        i. Definition
        ii. Properties
    - Cartesian product

2. Overview of functions
    - Function definition
    - Example and Counterexample
    - Function Range
    - Function Domain of Definition
    - Function Operations
        i. Addition
        ii. Multiplication
        iii. Division
    - Function Composition
    - Multivariate Functions
    - Parametric Functions

Application to Haskell

1. Types
    a. Basics types
        i. Bool
        ii. Char
        iii. Numbers
            1. Int
            2. Integer
            3. Float
    b. Some Operations on numbers
    c. Comparison operators
    d. Tuples
    e. List
        i. String
        ii. Generalisation
        iii. Some functions on list
        iv. List comprehension

2. Functions
    a. Declaration and definition
    b. Creating a simple Function
    c. Some useful constructs
        i. Conditional Expression
            1. *guard*
            2. *if [expression] then [expresion] else [expresion]*
            3. *case*
        ii. *let and in*
        iii. *where*

3. Higher order function
    a. Curried functions
    b. Some higher-orderism is in order
    c. Advanced function on List
        i. Map
        ii. Filter
    d. Function composition
    e. Parameterized Functions