

# Haskell Group Progress Report

Members:

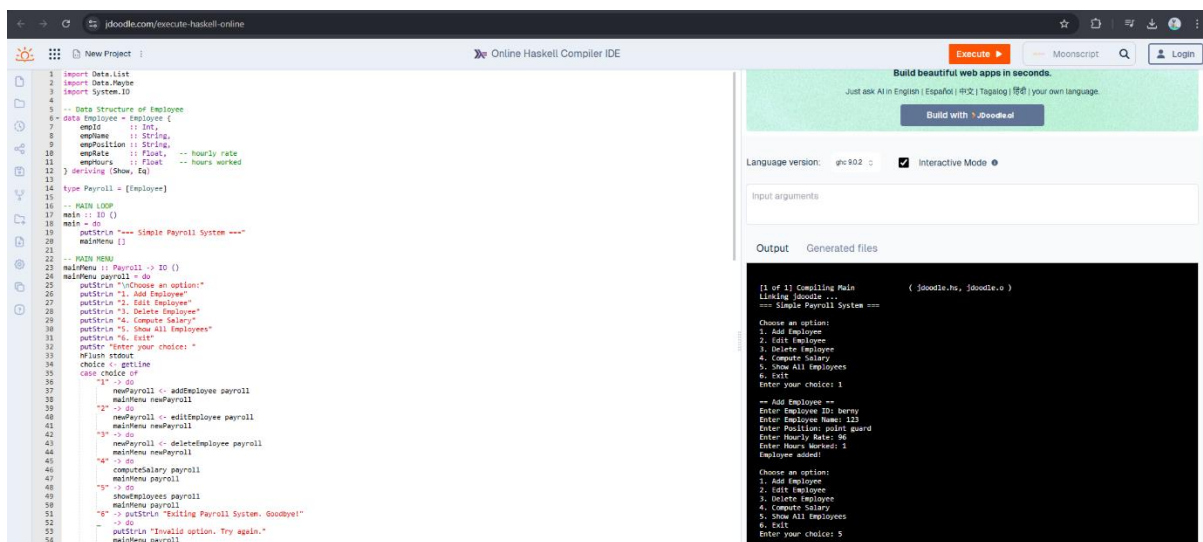
Del Mundo, Guiane Carlo

Lumba, Nelwyn Jairoh

Pugal, Reiven Curt

Rivera, Kurt Francis

This week, after learning the control flow structures of Haskell (using case statements), we have finished creating the main menu and submenu of our program. Below is a screenshot of the program and the inputs needed for the payroll system. Currently, it still lacks the database functionality as we are still researching of ways or plugins to link SQL and Haskell.



```
1 import Data.List
2 import Data.Maybe
3 import System.IO
4
5 -- Data Structure of Employee
6 data Employee = Employee {
7   empID    :: Int,
8   empName  :: String,
9   empPosition :: String,
10  empRate  :: Float, -- hourly rate
11  empHours :: Float, -- hours worked
12 } deriving (Show, Eq)
13
14 type Payroll = [Employee]
15
16 -- MAIN LOOP
17 main :: IO ()
18 main = do
19   putStrLn "=== Simple Payroll System ==="
20   mainMenu ()
21
22 -- MAIN MENU
23 mainMenu :: Payroll -> IO ()
24 mainMenu payroll = do
25   putStrLn "Choose an option:"
26   putStrLn "1. Add Employee"
27   putStrLn "2. Edit Employee"
28   putStrLn "3. Delete Employee"
29   putStrLn "4. Compute Salary"
30   putStrLn "5. Show All Employees"
31   putStrLn "6. Exit"
32   putStr "Enter your choice: "
33   choice <- getLine
34   case choice of
35     "1" -> do
36       newPayroll <- addEmployee payroll
37       mainMenu newPayroll
38     "2" -> do
39       newPayroll <- editEmployee payroll
40       mainMenu newPayroll
41     "3" -> do
42       newPayroll <- deleteEmployee payroll
43       mainMenu newPayroll
44     "4" -> do
45       computeSalary payroll
46       mainMenu payroll
47     "5" -> do
48       showEmployees payroll
49       mainMenu payroll
50     "6" -> do
51       putStrLn "Exiting Payroll System. Goodbye!"
52       return ()
53     _ -> do
54       putStrLn "Invalid option. Try again."
55       mainMenu payroll
```

Build beautiful web apps in seconds.  
Just ask AI in English / Español / 中文 / Tagalog / 日本語 / Your own language.  
Build with **1.doodled**

Language version: ghc 9.0.2 ☒ Interactive Mode

Input arguments

Output Generated files

```
[1 of 1] Compiling Main (Jsdoodle.hs, Jsdoodle.o)
Linking Jsdoodle ...
=== Simple Payroll System ===
Choose an option:
1. Add Employee
2. Edit Employee
3. Delete Employee
4. Compute Salary
5. Show All Employees
6. Exit
Enter your choice: 1
-- Add Employee --
Enter Employee ID: berry
Enter Employee Name: 223
Enter Position: point guard
Enter Hourly Rate: 96
Enter Hours Worked: 1
Employee added!
Choose an option:
1. Add Employee
2. Edit Employee
3. Delete Employee
4. Compute Salary
5. Show All Employees
6. Exit
Enter your choice: 5
Enter your choice: 5
```

Inputs:

**choice(String)** - input used by the user to navigate the program

**idStr(String)** - read input for employee ID

**name(String)** - input for employee name

**pos(String)** input for employee position in the company

**rateStr(String)** - input for employee's hourly rate

**hoursStr(String)** - input for hours worked of the employee

*Note: all read inputs in Haskell is in the String datatype, the program converts the String into the appropriate datatype for each employee information*

**Declarations:**

**Employee(Data Type)** – similar to class in Java, this sets up an object or a data type with the attributes empId, empName, empPosition, empRate, and empHours.

**Payroll(List)** – contains the list of employees in the program

**main** – is the entry point of the Haskell program

**mainMenu** – the function that contains the choices and the employee function options and control flow functionality

**addEmployee** – adds an employee to the payroll list

**editEmployee** – edits an existing employee's details

**deleteEmployee** – deletes an employee in the payroll list

**computeSalary** – computes the salary of an employee based on hours worked, hourly rate, and deductions

**showEmployees** – shows the employee database