COURSEWORK ASSESSMENT 4: INDIVIDUAL PROJECT

Assessed Learning Outcomes

This individual project assesses your knowledge of developing a concurrent system in Haskell.

Assessment Weighting: This component is worth 35% of your final mark.

Summary

Your task to simulate a concurrent "banking" system. The main program should spawn ten "customer" threads, and each of these threads model a bank account with a starting balance of £1000. The customers should then (at random intervals) choose one of the other customers (at random) and transfer a random amount of money (between £10 and £50) into their account.

Project Details

In this individual project you will be developing a concurrent "banking" system with 10 "customer" threads interacting with each other by transferring money between their accounts. You will find details of the project below:

- Your project should define an appropriate Haskell data type Customer, which should include their name, account number, and account balance.
- Your project should also contain a main thread which creates ten "customers" (ten values of type Customer), and spawns ten threads, one for each of these customers.
- Each customer thread should behave as follows: at random time intervals, the thread should select one of the other customers at random, and transfer a random amount of money (between £10 and £50) into their account. The amount transferred should not be more than what that customer has available in their account balance (account balances should not be negative).
- Your system should simulate 100 transfers, and then terminate and output the final account balance on each of the 10 accounts.
- Make sure each definition includes haddock style comments.
- You should also write a one-page report, detailing any issues you have faced, and justifying any design decisions you've had to make.
- Extra challenge (worth 20%). Making use of Control.Parallel, you could design your system in such a way that different customer threads actually run in parallel, on the multiple cores of your computer.

Submission Requirements

You should submit on QMplus your zipped Haskell source code together with the one-page report.

Marking Criteria

10%	Data type definition	Suitable data types defined, including making these members of appropriate type classes
40%	Implementation	Code meets the requirements given above, and works as expected
10%	Comments	Code is well-commented with haddock style documentation
20%	Report	The report clearly and concisely describes the design choices made, and any issues faced
20%	Parallelism	As an extra challenge, you should try to make use of Control.Parallel so that some of the customer threads actually run in parallel, on different cores of your computer

Submission Date: Wednesday 20th January 2021