Advanced Programming Coursework March 2020

Cycling Routes application

In this assignment you are required to develop a prototype for an application in Scala to extract information from cycling route data. You will be provided with cycling route data, and your application should allow the user to select from the set of options listed below and obtain the results of the chosen selection. The application should have a simple text-based, menu-driven interface.

Data

The data is supplied to you as a comma-separated text file *cyclingroutedata.txt*. Each line of the file contains the route name concatenated with the name of the starting point and 0 or more sets of stage values. Each stage set's values represent the stage number, the name of the stage endpoint and the length of the stage (in kilometres). A sample line from the file, edited for display purposes, is shown below:

```
Oor Wullie Route (GCU),1:City
Chambers:0.75,2:Velodrome:3.8,3:People's Palace:2.7 …
```

Note the stage values are separated by a colon.

Your application should read the file contents and store the data in a map structure where each line of the file is used to construct a map entry with the route name as the key, and a list of tuples as the value. The type of the structure should be ***Map[String, List[(Int,String,Float)]]***. The sample line shown should be represented within the map as follows:

```
Map("Oor Wullie Route (GCU)" -> List((1,City Chambers,0.75),
(2,Velodrome,3.8), …))
```

You will also be supplied with a fragment of Scala code, in a Scala Worksheet file *cyclingroutedata.sc*, that creates a map with the same format, containing equivalent data. You can use this, if you wish, to test other functions as required without the need to have completed the file reading functionality.

Options

Your application should allow the user to perform the following analyses:

1. Get all the route values and display suitably formatted.
2. Get the total distance and number of stages for each route, for all routes

   e.g. Oor Wullie Route has 6 stages and a total distance of 18.45km
3. Get the average total distance and average number of stages of all routes.
4. Get a report of all routes in descending order of total distance with a summary showing the overall average total distance and overall average number of stages for all routes.
5. Get the route details for a user specified route.
6. Allow the user to construct a personalised list of routes through selecting one or more routes and adding a comment to each chosen route.

Marks for each analysis will be awarded on the basis of completeness and correctness of the implementation.

Application structure

The application should be implemented as a Scala singleton object, with the cycling route data map as a field. Your code should firstly read the file data into the map. It should then display a menu, allow the user to input a choice and invoke a suitable handler function for that choice. The menu should have an option for each of the required analyses and also an option to quit the application.

For each analysis it is suggested that you define the following:

- A function that performs the required operation on the data and returns the results to be displayed
- A function, to be invoked by a menu option, that accepts user input if required for an operation, invokes the operation function and displays the results of the operation to the user

The functions for each analysis should be composed to perform the operation and display the result. For example, the first analysis listed above could be performed using:

- A function that is applied to the data and returns a result of type *Map[String, (Int,String,Float)]*
- A function that invokes the operation and iterates through the resulting map to display its contents.

Each analysis will differ in terms of the user input required and the type of result returned by the operation.

Marks will be awarded on the basis of completeness and correctness of the implementation, and on producing a well- structured solution.

Programming style

Scala doesn't impose a particular style of programming. However, in this assignment you are encouraged to prefer a functional style over an imperative style wherever possible in your solution. In particular, your solution may demonstrate examples of appropriate use of a range of the following functional programming techniques:

- Recursion
- Folding
- Pattern matching
- For comprehensions
- Partial function application

Marks will be awarded on the basis of evidence of your ability to implement functional solutions to problems across all aspects of the application.

Testing

You should test your operation functions using the test data - for each one you should define input data if required, note the result returned by the function and compare this to the expected result.

You should test your complete application by invoking each menu option using suitable user input and comparing the displayed result with the expected result.

Marks will be awarded on the basis of evidence of a rigorous approach to designing and implementing test cases.

Evaluation

You should evaluate the suitability of Scala for developing this application. Your evaluation should refer to language features and the support in the language for the programming approach that you have used in your solution and consider alternative approaches that you could have taken to solve the same problems either using Scala or using a different language that you are familiar with. Your evaluation should be 400-500 words long.

Marks will be awarded for demonstrating an understanding of programming models and their implementation in Scala, related to the requirements of the application.

Assignment Deliverables

You should submit the following:

Implementation:

• Your implementation should be submitted as an IntelliJ project folder contained in a ZIP archive

Documentation:

• A Microsoft Word document containing:

o Test plans and test results. Test results may be illustrated with screenshots

o Evaluation

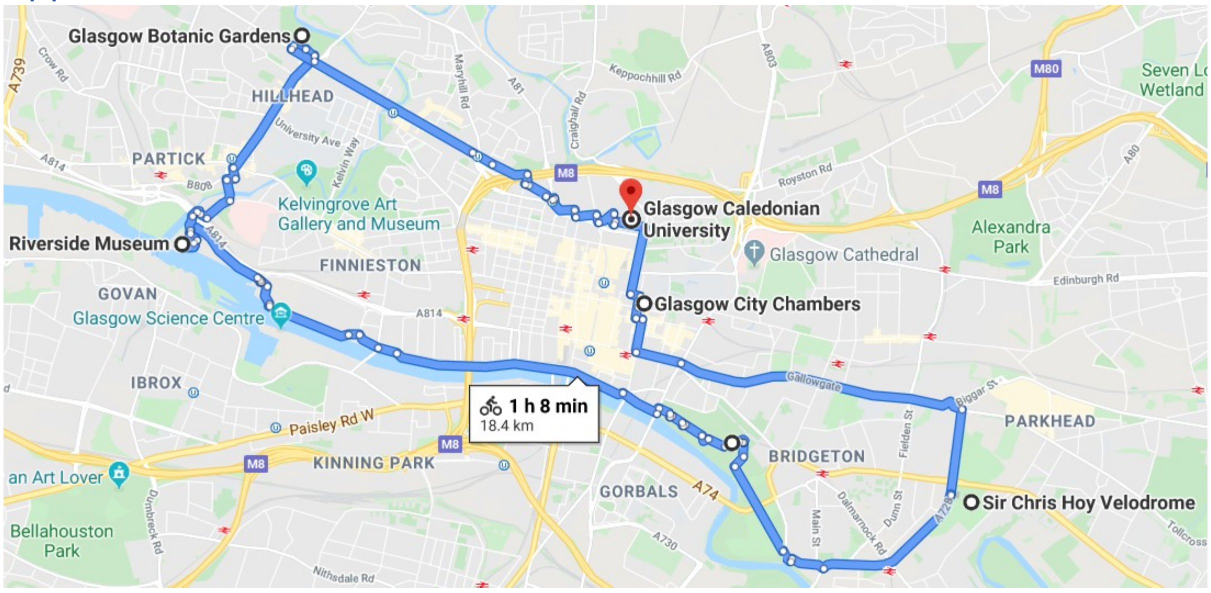
Submission: Thursday 9th April 2020 17:00 (tbc)

• Your implementation and documentation should be submitted in a ZIP archive via GCULearn. Submission instructions will be available on GCULearn.

Marking Scheme Options (operation & display)

All recorded values (Option 1)	**2**
Totals for each route (Option 2)	**3**
Averages (Option 3)	**3**
Ordered Report (Option 4)	**3**
Specified Route (Option 5)	**3**
Personalised Route (Option 6)	**6**
Application (file read and menu)	**10**
Programming style **(use of recursion, folding, pattern matching, for comprehensions, Partial function application etc.)**	**10**
Testing	**5**
Evaluation	**5**
Total	**50**

Appendix (Oor Wullie Route Details)



Route Details

Stage 1
GCU to City Chambers – 0.75km

Stage 2
City Chambers to Sir Chris Hoy Velodrome – 3.8km

Stage 3
Sir Chris Hoy Velodrome to People's Palace – 2.7km

Stage 4
People's Palace to Riverside Museum – 5.4km

Stage 5
Riverside Museum to Botanic Gardens – 2.4km

Stage 6
Botanic Gardens to GCU – 3.4km