|  |  |  |
| --- | --- | --- |
| Module Leader Chris Bates | | Level 6 |
| Module Name Functional Programming | | Module Code 55-601748 |
| Assignment Title Using and evaluating the functional paradigm. | | |
| Assignment type  Individual assignment | Weighting 100% | Magnitude  Completing this task should take around 30 hours. |
| Submission date/time  3pm on 5th January, 2023 | Blackboard submission: Yes Turnitin submission: No | Format  Zip file containing source code and report. |
| Planned feedback date  26th January, 2023 | Mode of feedback  You will receive written feedback on your submission through Blackboard. | In-module retrieval available No |
|  | | |
| Module Learning Outcomes  * Apply the functional approach to the design and implementation of software. * Evaluate the design of contemporary functional languages. * Evaluate the merits of functional programming, identifying where and how it is an appropriate development choice. | | |

# Introduction

In this assessment you will use Clojure to build solutions to a set of simple problems. You will apply the principles of functional programming such as immutability, recursion, the composition of functions, and lazy evaluation, and you will use Clojure best practices in your solution.

Additionally, you will write a short report in which you evaluate both the functional approach to programming and the Clojure language.

# The Problems

Clojure is used for many different tasks, but mainly it is used as a server-side language rather than for the development of user interfaces. Clojure code is robust, leverages the power of the JVM, scalable, concurrent, secure, testable and maintainable. In this task you will use many of these features, whilst also demonstrating that you understand something about the functional paradigm.

## One

Write a program that converts ASCII strings into Morse code and strings of Morse code into ASCII. Use the minimal standard character set that is the letters A to Z, the digits 0 to 9 and the space.

## Two

The Central England Temperature records (CET) is a series of daily temperature records that go back to 1772 and monthly records back to 1659. CET covers an area bounded by Bristol, London and Lancashire and forms the longest continuous temperature record in the World.

Download the relevant data files from [Met Office Hadley Centre Central England Temperature Data Download](https://www.metoffice.gov.uk/hadobs/hadcet/data/download.html) and load these downloaded versions in your program. The Met Office provides the temperature in two formats: the legacy format is the one that is described below; the 2.0.0.0 version files are simpler, but the data for 2019 to 2022 is in the legacy format. You may use either set of files but your code will be richer and more varied if you use the legacy file.

Using only the daily mean temperature records, answer the following questions:

1. Find the warmest day for each calendar month (e.g. the warmest January day, warmest February day and so on).
2. Find the warmest and coldest years.
3. Find the mean temperature for each calendar month (the average for all Mays, for example) and the instance of each month that has the greatest and smallest variation from that mean.

**Note** the format of the file is Column 1: year, Column 2: day, Columns 3-14: daily CET values expressed in tenths of a degree. There are 12 columns; one for each of the 12 months.

1772 1 32 -15 18 25 87 128 187 177 105 111 78 112

1772 2 20 7 28 38 77 138 154 158 143 150 85 62

1772 3 27 15 36 33 84 170 139 153 113 124 83 60

1772 4 27 -25 61 58 96 90 151 160 173 114 60 47

1772 5 15 -5 68 69 133 146 179 170 173 116 83 50

1772 6 22 -45 51 77 113 105 175 198 160 134 134 42

You may use the monthly and seasonal files to help in developing your code but not in performing your final calculations.

# The report

You must write a report in which you address the following question:

**Evaluate the functional paradigm of software development, comparing it with imperative, object-oriented development.**

The report should:

* Be no longer than 800 words.
* Must include citations and references of any sources that you use. Sources that require citations include academic papers, blog posts, discussions on forums such as Stack Overflow, videos, podcasts, and textbooks.
* Include a URL that I can use to access your video presentation (see below).
* Be in PDF format.

Help with writing academic reports can be found at:

* <https://libguides.shu.ac.uk/academicwriting>
* <https://libguides.shu.ac.uk/c.php?g=684327&p=4886064>
* <https://libguides.shu.ac.uk/referencing>

# The video

To demonstrate your code, you must create a screencast with a maximum duration of 15 minutes, in which you show the code that you have written, execute it, and demonstrate any tests that you have created. The video is an opportunity for you to show that your solution meets the brief, that you understand how it works, and for you to discuss why you built it as you did.

Do not upload your video to Blackboard. Place it in a suitable repository such as your Google Drive, OneDrive or YouTube. Get a sharing link that lets me view the video online and in my own time, and put the link in your report.

# Submission

You must submit to Blackboard a single zip file that contains your source code and your report. The deadline for this task is **3pm on 5th January, 2023**.

Submit your Clojure code as a Leiningen-compatible project. If you use an IDE such as IntelliJ IDEA or VS Code, simply archive and submit the whole project, but not any libraries. You will demonstrate the application in your video which means that I may not need to execute your code. I will, though, be reading it.

You **should** use specifications where appropriate, and **must** include meaningful, descriptive comments before key functions.

If you use algorithms, code samples, or inspiration from sources such as Stack Overflow discussions, you **must** include a citation in your code as a comment. You are expected to use a range of source material of your own when developing your solution. Doing so without citation is plagiarism, doing so with a citation is good practice.

# 

# Marking Scheme

Your work will be marked using the following scheme. Each aspect will be graded using the scheme which is outlined at <https://tinyurl.com/ytrnnf6e>. Your final mark will be normalised to one of those in the University’s level 6 common grade descriptor which can be found at <https://tinyurl.com/n46wtdua>.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Marks | At a pass level you will... | At a first class level you will... |
| Question one | Worth 20 marks | | |
| Completeness of the solution | 5 | * Some of the required functionality is implemented. | * All the required functionality is implemented. * The solution works as expected. |
| Quality of the code | 10 | * The code produces answers but these may be error-prone * Code structures such as lists and functions are generally used in appropriate ways * There may be places in which other structures would be better * Functional techniques including recursion and pattern matching are used sparingly | * The code is recognisably “functional” throughout * Clojure language features such as spec are used * Functional techniques are used appropriately throughout. * There are few, if any, places in which different code structures would be a better fit |
| The use of specifications and unit testing | 5 | * Write a few unit tests. * Attempt to use specifications. | * Write tests that exercise a variety of aspects of the code. * Provide specifications for, and tests of, those functions that handle external data. * Write specifications and tests for functions that are used/re-used throughout the application. |
| Question two | Worth 40 marks | | |
| Completeness of the solution | 10 | * Some of the required functionality is implemented. | * All the required functionality is implemented. * The solution works as expected. |
| Quality of the code | 20 | * The code produces answers but these may be error-prone * Code structures such as lists and functions are generally used in appropriate ways * There may be places in which other structures would be better * Functional techniques including recursion and pattern matching are used sparingly | * The code is recognisably “functional” throughout * Clojure language features such as spec are used * Functional techniques are used appropriately throughout. * There are few, if any, places in which different code structures would be a better fit |
| The use of specifications and unit testing | 10 | * Write a few unit tests. * Attempt to use specifications. | * Write tests that exercise a variety of aspects of the code. * Provide specifications for, and tests of, those functions that handle external data. * Write specifications and tests for functions that are used/re-used throughout the application. |
| The report | Total of 30 marks | | |
| Evaluation of the use of functional programming | 15 | * Outline some features of the paradigm. * Take all of your ideas from taught material. * Provide lists of “good” and “bad” points. * Give a supporting example. | * Discuss what is meant by functional programming and show how it differs from the imperative paradigm. * Go beyond the taught material to find your own sources. * Cite your sources throughout. * Establish your own evaluation criteria, and use them in making your argument. * Use appropriate supporting examples throughout. |
| Discussion of the Clojure language | 15 | * Describe Clojure. * Compare Clojure to another language that you know. | * Provide a detailed description of the language. * Show how the language is used to implement complex software. * Compare Clojure to another functional language and to an imperative one. * Show how the language integrates with Java libraries. |
| The video | Total of 10 marks | | |
| Demonstration and explanation of your code | 10 | * Create a video which shows you using the program. * Show little of the code * Give explanations that are weak, incomplete, or incorrect. | * Make a video in which you clearly show the program being used. * Show and explain important or interesting sections of code. * Explain how the code works and why you wrote it as you did. * Discuss the ways in which your system handles data. |