# Predicting Weather Data for Cambridge, UK

Michael Trapp
ACSE5 - Assignment no. 1

January 27, 2019

**Code Design**. The code follows an object oriented style with a flat hierarchy. There are two classes: "IOStuff" and "Prediction". The first class deals with input and output, while the second class handels the computation of the data. Class objects are created in the main method, which subsequently calls class methods to modify these objects. The objects are always passed by reference, as all methods are of "void" return type.

**Structure.** The program is divided into three files: First, a header file "Definitions.h" with type definitions, as well as class and function declarations. Second, a .cpp file with function definitions and third, a .cpp file which contains the main method.

**Input-Output.** The class "IOStuff" contains variables such as multimaps and filenames, which are assigned to data from input files. These data assignments are executed by means of class methods. The "IOStuff" class contains three methods, which are called subsequently from the main method. The first method "readinput" reads user input from a user input file with fixed name. The user conveniently writes the input in a text file prior to execution. The next method is "readdata", which reads the tabulated data from another text file, specified by the user. The data file name and content is variable, but must follow a fixed column structure. The data for this assignment was assumed from the website: "https://www.metoffice.gov.uk". The data is read line by line, with regular expressions distinguishing between comments and data content. The data is stored in two multimaps. The multimaps match keys, here either months or years, to values, here a data structure containing temperatures, rainfall and an arbitrary amount of other information. The third method "writedata" writes the predicted data into output files. This includes the output file "CambridgeOutput.txt" for the user and one output file for each predicted year for gnuplot to plot.

**Execution**. For execution, the following files must be present: "UserInput.txt", "Functions.cpp", "Source.cpp" and some data file, e.g. "CambridgeInput.txt". The main file "Source.cpp" can then be easily executed using a C++17 compiler. The program was successfully tested with Visual Studios 2017.

**Strengths**.

a. Although including input with the "cin" keyword was considered, it is deemed not practical, as multiple program executions would be too time-consuming. The applied method of user input via an input file offers a great deal of automation and convenience. Gnuplot was applied to automatically visualize the data.

b. The implementation of the prediction part is done in such a way that the functions can be defined generic. The code is very modular and allows for easily exchanging functions and methods. This is very useful if the program would be extended further. The output file is structured nicely in such a way that the data can easily be processed further.

c. The code is sustainable. It contains several assert statements to verify the correctness of the procedure. Messages in the terminal contain useful information for the user.

**Weaknesses**.

a. Although structures appear to be convenient for storing the multimap values, traversing a structure is identified to be most difficult and alternative approaches could be considered for future projects. However, a nice work around with switch cases is implemented.

b. Only 2 prediction methods (Lagrange Extrapolation and Linear Regression) are implemented. A Fast Fourier Transformation (FFT) was considered, as it has been a successful approach in previous course work. However, c++ appears not to have an inbuilt FFT algorithm, and Imperial College staff appears not to be familiar with this method for support and advice. However, the implementation of the ”year” and ”month” multimaps (month:data and year:data) allows for additional generic methods to be easily included, had the project been extended.