 Brady Code Challenge

Junior Software Engineer role

V1.1

September 2018

Brady PLC

# The Code Challenge

**Overview**

The Code Challenge is a part of the Brady recruitment process, and will help us determine your level of competence. You are strongly encouraged to approach it with a professional attitude and deliver a working, **production – grade solution**. Please apply OO design principles where appropriate. The submission is considered invalid if it does not include the full source code.

In case you use external sources, libraries or research, be sure to include your sources in an accompanying document along with any assumptions you make.

Please keep in mind that this code will be reviewed and evaluated, and we expect you to be able to present it, execute it and possibly even change it during your interview.

Once you have completed the challenge, please email it to [Ross.Jelf@bradyplc.com](mailto:Ross.Jelf@bradyplc.com). You can also provide a link to GitHub repository or other cloud storage locations.

**Do not include any .exe** files that would potentially prevent delivery via email. A confirmation of receipt email will be sent accordingly.

**Description of the Task**

You are required to provide a production ready solution (Console Application) written in C#, Java, C++, Python, Ruby, Objective-C, and any other [object oriented language](https://en.wikipedia.org/wiki/List_of_object-oriented_programming_languages) that you feel most confident in for the following requirements.

An XML file containing generator data (see accompanying file *GenerationReport.xml*) is produced and provided as input into an input folder on a regular basis.

The solution is required to automatically pick up the received XML file as soon as it is placed in the input folder (location of input folder is set in the Application *app.config* file), perform parsing and data processing as appropriate to achieve the following:

1. It is required to calculate and output the **Total** **Generation Value** for each generator.
2. It is required to calculate and output the generator with the **highest** **Daily Emissions** for each day along with the emission value.
3. It is required to calculate and output **Actual Heat Rate** for each coal generator.

The output should be a single XML file in the format as specified by an example accompanying file *GenerationOutput.xml* into an output folder (location of output folder is set in the Application *app.config* file).

Note that the provided XML examples are for the illustration only. They demonstrate the expected XML structure but do not contain the matching data. For sake of clarity *GenerationOutput.xml* does not provide a solution for *GenerationReport.xml* included in the package.

**Calculation Definitions and Reference Data**

**Daily Generation Value** = *Energy* x *Price* x *ValueFactor*

**Daily Emissions** = *Energy* x *EmissionRating* x *EmissionFactor*

**Actual Heat Rate** = *TotalHeatInput* / *ActualNetGeneration*

Emissions only apply to fossil fuel generator types e.g. coal and gas. There could be a varying number of generators of a given type in any one *GenerationReport.xml* file that is produced.

**ValueFactor** and **EmissionsFactor** is static data sourced from the accompanying XML file *ReferenceData.xml*. Note: it is not possible to change static data while the console application is running.

**Generator Types** map to **factors** as follows:

|  |  |  |
| --- | --- | --- |
| **Generator Type** | **ValueFactor** | **EmissionFactor** |
| Offshore Wind | Low | N/A |
| Onshore Wind | High | N/A |
| Gas | Medium | Medium |
| Coal | Medium | High |