OCR ERL Interpreter Project

Computer science NEA – PATRICK WILLIAMS (8123)  
Saint olave’s Grammar School

# Contents

## Analysis of the problem

* Problem identification. *(Establish Problem and Stakeholder)*
* Problem Research. *(Similar Problems and Analysing their Solutions)*
* Proposed Solution. *(Basics and Stakeholder Needs)*
* Solution Research. *(Technologies Used and Computational Methods)*
* Features of the Solution. *(Essential Features, Limitations, Requirements, Success Criteria)*

## Desgin of the solution

* Decompose the problem.
* Describe the solution.
* Describe the approach to testing.

## Developing the solution

* Iterative development process.
* Testing to inform development.

## Evaluation

* Testing to inform evaluation.
* Success of the solution.
* Describe the final product.
* Maintenance and development.

## Final codE

* Source code.
* Version History.

Analysis of the problem

# Problem Identification

## Establishing the Problem

All programming code given in the OCR Computer Science GCSE will be presented using the OCR Exam Reference Language (ERL), so it is important that the students have a solid understanding of the syntax of the language in order to be able to successfully answer many of the questions. Additionally, for questions where the student is required to write their own code, students are given the choice to respond with either the ERL or a high-level programming language.

Despite the code-based questions being written in ERL, a large proportion of GCSE students choose to answer writing questions with a high-level programming language instead of the ERL because of how unfamiliar they are with OCR’s Language. This brings up the question of whether students in school should be primarily taught with the ERL or a high-level programming language of the school’s choosing.

### Which option is better?

It is of course important that students are taught the basics of a high-level programming language in school, due to its real-life application in their future. However, in terms of the exam papers themselves, there are advantages in using the ERL instead of an alternate language.

#### syntax differences

The syntax of the ERL differs in some ways from high-level programming languages, which can create confusion during the exam. Although the exam board is lenient, allowing small changes from the formally defined ERL syntax and instead focusing on the logic behind the code, there are some large differences which can cause confusion and for marks to be lost in an exam.

🡺 INCLUSIVITY AND EXCLUSIVITY  
One difference between ERL and many high-level programming languages is very prominent with counter-controlled loops. Take these code snippets below:

|  |  |
| --- | --- |
| A black background with white text  Description automatically generated |  |

A screen shot of a computer code

Description automatically generatedDespite both using the same numbers, ERL will print “Hello World” 4 times, whereas Python will only print it 3 times. This is due to ERL being an inclusive language, which therefore can create confusion with students using Python, a very popular choice, and can cause errors throughout the paper.

This issue is not unique to Python, in languages with C-like style of creating for loops, including Java, PHP and JavaScript, the standard is to create for loops with a < operator, rather than <=, therefore also being naturally exclusive. With ERL being a rare case of inclusivity, this can easily cause issues, especially with “fill-in-the-gaps” style questions and for students who attempt to use ERL without a thorough understanding of it in writing questions, where this difference can easily lose a student marks.

🡺 SUBSTRINGS  
There is a drastic change to the formatting of producing substrings in ERL compared to the standard for many other programming languages. This can also be a source of confusion in the exam.

|  |  |
| --- | --- |
|  |  |

The OCR ERL has an unorthodox substring syntax, with the first argument remaining as the initial index, but the second argument being the length of the returned substring in characters, instead of the standard of the final character position +1. Again, an unfamiliar student may get caught out in an exam.

🡺 MISCELLANIOUS CHANGES  
There are also many small, but still potentially dangerous changes between languages.

* The use of “MOD” and “DIV” instead of symbols in typical languages
* The use of ^ as an exponent (\*\* is used in Python, a very commonly chosen language)
* .upper and .lower are properties rather than methods (have no brackets afterwards)
* The use of closing keywords such as endif and endswitch (uncommon in modern languages)

Although many of these minor changes will generally be overlooked by an examiner during the marking process, a student who is fluent in the syntax will feel much more comfortable in the exam when occurring the custom syntax which they might be unfamiliar with.

#### Learning Resources

A close-up of a book

Description automatically generatedA paper with text and a questionnaire

Description automatically generatedEven though it can be argued that a high-level programming language can be interchangeable with ERL in the exam itself, many of the textbooks which are given to students use ERL in their explanations and exercises for students to do. This is as there is no set high-level language that any given school teaches, and the studied language is chosen by the school itself. Therefore, ERL is used within resources as a universal language utilised by the majority of textbook. This means many programming concepts, such as sequence, selection and iteration, as well as algorithms, such as linear search and bubble sort, will have their code written in ERL and this is how students will encounter and learn these key concepts for the exams.

This therefore presents ERL as a superior language than any alternative for the OCR GCSE, due to its syntax aligning with the mark scheme, and its wide usage within resources which are taught to pupils.

### WHy are alternative languages more commonly used?

The main issue with the ERL is the lack of a way for students to use it. ERL is not a pseudocode, and is an equivalent to a high-level language, so has been written in a way where it is interchangeable with real, functional code. However, there is no translator available which allows students to write and consolidate their skills with the ERL. This alone makes using the ERL mainly redundant, as students can only become familiar and comfortable with a language after using it for a prolonged period, as it this process of trial and debugging that allows their code to be reliable and ensures that they can produce accurate and correct code during the exam.

Therefore, our problem is that students have no way to run ERL to practice before the exam, in order to allow students to be more comfortable in an exam.

## Stakeholders

A solution to this problem would, logically, be useful to the students who are taking the exams, as well as being used by a teacher in a lesson to aid the students.

### Current gcse student

F

### past gcse student

A

### Computer Science gcse

A

We will return to our stakeholders later to ask for their needs in a potential solution.

# Problem Research

## Solutions for other languages

There is an abundance of interpreters available for other languages, and I will be analysing them based on their application towards our stakeholders as alternatives, not for their typical users.

### NOde.JS in Terminal

This is a command-line interpreter for JavaScript which can be run in a terminal on a device by calling it with “node” with no separate application.

#### Advantages

A computer screen shot of a code

Description automatically generated🡺 The obvious first advantage, is that it successfully runs JavaScript code when entered. This may sound irrelevant to mention as a solution, but as there is no option for ERL that does this, it already presents this interpreter as much superior to any ERL methods that could be chosen.

A screen shot of a computer

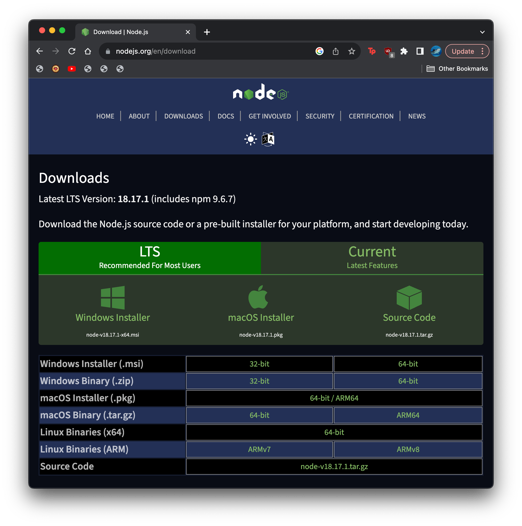
Description automatically generated🡺 It also contains error handling, referencing both the error type and where it occurred, which is very useful for anybody writing code, especially to students who are prone to mistakes, making learning the language much easier as well as aiding debugging.

A screenshot of a computer error

Description automatically generated🡺 Multiple use modes allow the students to use the program depending on their needs. The shell, for a few lines of code to be run individually; the editor for when a longer program is required, to all be interpreted as once; and the ability to select a file in local storage and run it. This makes the interpreter much more diverse and can adapt to the need of the user.

🡺 The basic help feature allows the user to be able to fluently use the compiler, as a reminder on how to operate it with the command-line interface.

#### Disadvantages

🡺 Lacking a graphical user interface of any type, this makes it very difficult to use, especially for the stakeholders who are first-time programmers. A command-line interface is very complex, and the terminal is often blocked on school devices, making it unapplicable for our student stakeholder who need to use it.

🡺 Installation is also difficult, requiring installation from the command line or downloads from the website, meaning a student requires an administrator to be able to access it, which reduces the scope of the interpreter.

🡺 The poor user experience stems from the command-line interface. Users must memorise many commands to access the different features of the interpreter. Although this makes it much faster to use for experienced programmers, for beginners it becomes much more tedious.

🡺

#### what can i apply to my solution?

### Python Idle

### Replit

## Solutions FOR Erl