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 practice ERL, so are unfamiliar with it by the time the exam occurs.**

## Stakeholders

The target audience for our solution would be, obviously, for students partaking in the OCR Computer Science GCSE, however more specifically, they would likely be:

* First time programmers, with little or no prior programming experience.
* Using restricted school devices or poor-performance home devices.
* Unable to install and setup any complex software.
* Require a very intuitive solution with lots of help at disposal.

In order to gain a greater insight into the experiences of potential stakeholders, I have interviewed 3 different individuals to understand their experience with programming. I will also return to these stakeholders later to ask for their needs towards a solution I decide on.

### **current gcse student**

* Past experiences with programming?
* Familiar with the ERL?
* Plans for revision?

### **past gcse student**

* Did they use ERL or high-level in GCSE?
* How did they practice using ERL?
* Find the experience easy/difficult?
* Potential solution would have aided revision?

### **Computer Science teacher**

* Programming focused on high-level or ERL?
* Limitations on using ERL?
* Potential solutions be useful towards teaching?

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

# Problem Research

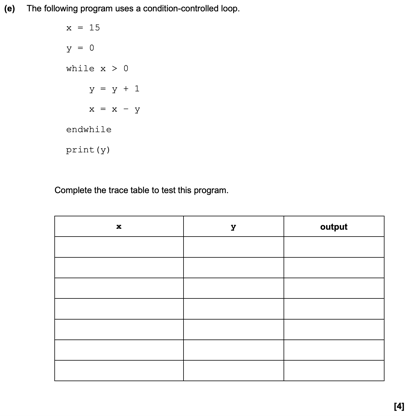
There are many other problems with pre-existing solutions, which can be analysed to provide useful insight into what my potential solution can be.

## Solutions FOR Erl

Initially, the focus will be on pre-existing ways that students can run ERL code.

### **trace tables**

A white sheet with black text

Description automatically generatedStudents should already be familiar with trace tables, as there are questions in the exam papers, for example this one from the sample paper, of students being required to use them to “run” ERL code. However, when used to run a student’s own code, it presents a slightly different task.

#### Advantages

🡺 Students are taught about trace tables, so it is an easy and familiar method for them to use on their own code, and it allows them to check if the logic behind their program is functional.

🡺 It serves as a form of debugging, as it allows students to analyse the logic behind the program that they have written and allows them to track the values of the variables throughout – much like a debugger – which can help produce more accurate code.

🡺 It does not require any software, just a pen and paper so it is a very feasible solution which can be done anywhere, making it very accessible.

#### disadvantages

🡺 There is no syntax checking at all, so whether the code is accurate cannot be determined as it is down to the students understanding of the language and may be incorrect in an exam.

🡺 This is also true of the logic, as if a student has a flawed understanding of how the logic of the language works, as they are the interpreter in this scenario it means that they may unintentionally gain an incorrect solution which they believe to be correct, as they have interpreted it based on their inaccurate knowledge of the language due to their unfamiliarity, and would lose them marks during an exam.

🡺 The task is very slow and tedious, especially when it comes to longer programs, so there becomes a point where it is intangible to use a trace table when the program reaches a certain size, because the number of variables becomes too tedious to track, and a much faster solution would be preferable.

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

Although, overall, a trace table is a horribly unreliable and slow method of running a program, it does teach lessons about ease of use which are very important to my own program, in order to make it available to beginners. Also, the very accessible nature is of importance, as there is no use of coming up with a better solution if it is unavailable to a vast number of students.

### **translation to other languages**

This method involves a student writing the original code in ERL, then translating it to a high-level language of choice in order to run it and obtain a result.

#### Advantages

🡺 Does allow for the program to be ran, and the student can produce a result, which allows students to test programs and feel comfortable with their ability.

🡺 Allows simultaneous learning of ERL and a high-level programming language at the same time, widening the scope of the student and producing a more educated individual.

🡺 Having to re-read ERL to translate it allows the student to scan for errors more effectively, hence making them more accurate with the language,

#### Disadvantages

🡺 The ERL stage can become heavily redundant, with very minimal thought behind it, as the high-level programming language is what is ran, so the student would tend to become much more proficient in that, as the ERL was just an unnecessary stage which can become ignored and therefore not learnt by the student.

🡺 Very minimal focus is applied to how correct the ERL written is, as it is only the high-level language that is executed, so if a student makes lots of errors in the ERL, but produces a correctly functioning high-level code, it can result in them learning incorrect syntax for ERL in cases. This can make this method negative, as the false reinforcement could lead to very big flaws, as it is only their own interpretation of the syntax of the language which determines the result of the program.

🡺 Again, the process is very time consuming, and really discourages the student from attempting to write accurate ERL as it makes the programming process so much longer.

#### What can I apply to my solution?

The very negative evaluation of this method shows the importance of a fast solution that encourages the student to write in ERL, as otherwise using a different high-level language would be a much easier solution for them.

### **Conclusions**

The overall result of this analysis is that an interpreter for the ERL is required, as otherwise running ERL is such a time-consuming process for the student. This would require a program that would be able to be presented a program written in ERL, and directly produce a result from it, as otherwise students would be much better off with using a high-level language.

Therefore, researching how interpreters for other high-level languages function would be beneficial in identifying which key features are important for me to implement.

## 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

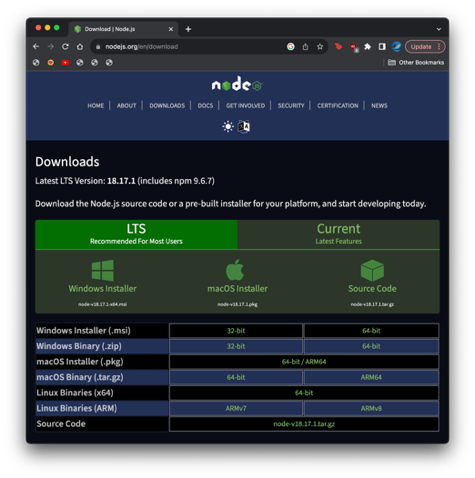
Description automatically generated🡺 It also contains error handling, referencing both the error type and where it occurred, which is very useful for anybody writing code, especially to students who are prone to mistakes, making learning the language much easier as well as aiding debugging.

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, and could do with a much more intuitive solution, as well as the shell being slightly clunky to use at points,

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

🡺 Ensure that error handling is implemented as it is very important in an interpreter  
🡺 Provide an intuitive interface as the interpreter is intended for beginner students.  
🡺 Prioritise accessibility and that it is available to many.

### **Python Idle**

I will be building upon the previous analysis of node.js for this analysis, as many of the basic features, such as running and error handling are taken for granted, so will not be mentioned.

#### Advantages

A screen shot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a phone

Description automatically generated🡺 The user experience is vastly improved, with use of the ribbon to introduce a greater array of options that the programmer can use, allowing equal if not greater functionality than node without the need to memorise all the commands, making it vastly easier to use for beginners.

🡺 Includes a debugger, with persistent breakpoints, stepping, and viewing of global and local namespaces, which can aid the students understanding of the language as, after all, it is correcting mistakes that allows a student to develop their understanding of a language.

🡺 Overall a more easy-to-use solution. Some examples being the syntax highlighting making it easier to identify blocks of code, as well as the shell element of the program being very logical to use, which runs after the program is complete, even allowing the previous file that is run to be referenced from it, making testing very easy. Overall, the interpreter is very well made and intutive for beginners to use.

#### A screenshot of a computer Description automatically generatedDisavantages

🡺 There are still accessibility issues present. The program still needs to be downloaded from a website or using a package manager, and will still require administrative permissions to install, which can decrease the scope of the program in terms of who can use it.

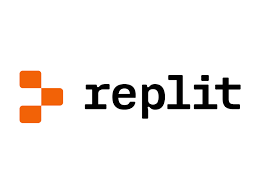
🡺 In terms of helping absolute beginners, the official documentation is still quite dense and hard for new programmers to undestand. Although there are a multitude of beginner guides avliable on the internet, when thinking soley about the official documentation, it would make the experience harder and is worth considering for an ERL interpreter which is aimed directly at new programmers.

🡺 Having to save files locally before being able to run them can be a nuisance a points: a very small critique but worth considering, as overall IDLE is a very strong interpreter.

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

Trying to replicate the visual, ease-of-use of IDLE is a strong target, as it makes focusing on the code itself very easy, as operating the interpreter itself is very straightforward. In terms of improvements, the focus is still towards making it available to as many people as possible, so considering low-performance devices and avoiding downloads would still be useful.

### **Replit**

Replit is an online IDE which allows people to write code on a website, where it is then run on their cloud servers and the results are returned to the client.

Again, the evaluation builds upon the points of the previous, as many of the advantages are shared with IDLE, including debugging features and syntax highlighting.

#### Avantages

🡺 The accessibility is the highest out of any solution. Being a website, it requires no installation, so can be run by any student without requiring assistance from an administrator. Additionally, the program itself is ran on cloud servers, so any low-performance devices are not affected due to the program being executed externally, resulting in the solution having a very large scope.

A screenshot of a computer

Description automatically generated🡺 The community aspect is extremely prevalent, with an extremely large number of users sharing their code, and aiding others by providing advice and feedback. This positive loop leads to a very good environment for new coders in order to learn programming.

#### Disadvantages

🡺 A potential negative is that the functionality of the solution has resulted in the interface becoming very convoluted, and potentially confusing to a new user, which could make programming become a much less approachable experience for them.

🡺 It doesn’t run ERL!

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

A website solution seems extremely promising, as it allows it to be extremely accessible to lots of students. Due to the huge size behind Replit, some features such as the cloud servers and community feedback will be impossible to replicate, but the focus on creating an environment where anyone can run code, with a focus on learning is a strong takeaway.

# Proposed solution

The obvious solution to our problem is to create an interpreter which can run ERL to allow students to practice using the language.

## Stakeholder Needs

### **Current gcse students**

### **past gcse student**

### **computer science teacher**