analysis

# project identification

In the OCR Computer Science GCSE, all programming code will be presented using the OCR Exam Reference Language (ERL). This is their equivalent to a high-level programming language and is designed to have simple and pseudo-code like syntax to be widely understandable.

Especially in paper 2, it is vital that the students understand the language as they will need to be able to understand programs written in it, as well as completing sections of it in larger template programs, as well as being given the option to write in it for the longer “Write” questions.

Unfortunately, students become familiar and comfortable with a language through usage, and the large majority of students lean towards using other high-level languages – primarily python – throughout their paper due to a lack of usage prior to the exam. Therefore, to help benefit GCSE students, I intend to create some form of translator, which would allow students to write ERL code, and for the program to then translate and execute that code for them in order to allow for practice.

## issues with high-level languages

A black background with colorful text

Description automatically generatedA black background with white text

Description automatically generatedOCR has attempted to design the papers in a way that ERL can be used interchangeably with other languages, however there are differences in the syntax which may lead to confusion and a potential loss of marks in the exam.

🡺 Inclusivity and Exclusivity

A screen shot of a computer code

Description automatically generatedWhen comparing the two code snippets on the left, they look very similar, but the ERL will iterate once more than Python – the more familiar language – as its counter-controlled loops are inclusive rather than the typical exclusive.

This may seam unique to Pythons range() function, but in more standardly defined loops, such as in C, it is the equivalent of ERL using <= rather than <, which is expected, therefore potentially causing confusion during the exam .

🡺 Differing feature-sets

This, again, vastly changes depending on the language. For example, a regular user of python may be confused by the premise of a procedure compared to a function, as well as being told to use a switch statement potentially not being in their vocabulary.

The worst instance of this, however, is with in-built methods. Only certain standard methods are allowed, such as length and string manipulation methods. However, if a student who is used to python decides to use min(), max() or .sort() in their written example, then they will be awarded no marks for that section of the program as they have shown no coding ability of knowledge. Being familiar with ERL, even if not using it for “Write” questions, will help educate students on which methods they can and cannot use.

🡺 Small syntax changes

These changes most likely would not lose the student marks in “Write” questions, but if the students are unfamiliar with them may cause confusion when reading ERL algorithms.

A black background with white text

Description automatically generatedA black background with green and orange text

Description automatically generatedFirstly, the typical parameters for the start and end position of a substring has been changed to the start and length in ERL, which is important to recognise.

Some other small syntax changes include:

* The use of closing keywords such as endif and endwhile (not frequently used in modern languages)
* The use of MOD and DIV rather than symbol equivalents which they may be used to
* The use of ^ as an exponent, compared to \*\* which is more frequently used.

🡺 Learning resources

Even though it could be argued that a high-level language could be used in the exam, the vast majority of textbooks use ERL in all their examples due to the fact that there are differences in all high-level languages, and it exists as a means to link together the GCSE students. Therefore, if a student is primarily learning how to code whilst using ERL, it makes sense to practice with it too.

## the solution

It is clear that the ERL is important to understand, so therefore a translator would help students complete practice questions – where they are written in ERL – and actually have a result to check and ensure that their solution works. Furthermore, it would help to transfer skills between different languages, as a semi-experienced programmer could test programs in ERL to see the transferability of their skillset into the exam. Finally, for complete beginners, it provides a way to run this very English-like language in order to introduce them properly to coding for the first time without the complexities of other languages.

It will also benefit teachers, as it will become much easier to explain the ERL to students and allow them to practice in lessons as it aligns with their teaching resources and is a much more logical approach to the exam. It would also be very useful for demonstration, as a key element of programming is the thought process behind it, so showing the class how to construct more complex algorithms – which at GCSE are generally sorting algorithms – alongside an actual result which also allows for easy debugging.

# stakeholders

There are two primary groups who would be utilising the solution: students and teachers, and there are generic descriptions for both of these groups.

|  |  |
| --- | --- |
| Students | Teachers |
| First time programmers with minimal experience | Knowledge of programming and ERL |
| Using restricted or low-performance devices | No restriction on software downloads |
| Difficulty installing complex software | No issues with installing software |
| Require an intuitive solution with help available | Still require basic error information |
| Would be helpful to include learning features | Solution must be presentable and teachable |

I will specify further with my stakeholders as to the exact details, but overall, a simpler solution seems to be ideal, as a translator which includes too many settings or features seems overly complex for new beginner programmers who are just trying to write very basic programs.

In terms of stakeholders, I have a current GCSE student who is the target user, a past GCSE student who has experience using the ERL and revising with it, as well as a computer science teacher in order to see how a possible solution could be implemented into a classroom.

## beginner programmer

A person writing on paper

Description automatically generatedAbigail Williams is a student in year 10 who has been studying computer science in some form for 3 years but has only begun learning the OCR GCSE specific content this year. I asked her questions about her computing background.

🡺 What are your past experiences with programming?

A child typing on a computer

Description automatically generated“My previous experiences with programming have been rather bad because I struggle to understand the computing language format and how it all goes together. On top of this, it takes me a long time to type it out in the language, and I would prefer it was simpler and more like English.”

🡺 What’s your experience with the Exam Reference Language?

“What’s the exam reference language? I have used python before in the past, but I have never heard of this one until like 5 seconds ago.”

**(After an explanation)** “Okay, that seems important to know if it’s going to be used in the papers, so I should probably make sure I learn about it a little bit”.

🡺 What are your plans to revise coding?

A child pointing at a piece of paper

Description automatically generated“I’m going to make a revision booklet and write notes in it about the rules of the language that I’m going to use so I can try and remember. I know practicing is meant to be good, but I don’t know how to, and it sounds very difficult so I’m not sure if I will.”

**🡺 Conclusions 🡸**

The key element of this is her very small amount of experience with programming. Although this may reduce how useful her feedback is during an iterative development process, it is important to consider that she is the main user of the program, so later on her ideas for usability should be very constructive.

## past gcse student

A child in a suit and tie sitting at a table with a computer

Description automatically generatedTanish Arjaria is a year 12 student who did OCR GCSE Computer Science last year and achieved a grade 9, who is now continuing to take Computing for A-Level. He is therefore a very high-achieving student, ideal for providing feedback about the process behind building the translator, however he is not a realistic end user as he is past the point of learning about ERL, and no longer has a need to use or practice it, but his experience will be useful.

🡺 Did you use ERL or a high-level language in the GCSE?

A person sitting at a table with his head on his face

Description automatically generated“I chose to write in python in the exam papers. Although I know a decent bit about ERL, I am not as confident in using it as I am with python, so I felt as if it was a less risky choice. However, for the questions where code had already been written in ERL, such as the fill in the gap ones, I did write in ERL.”

🡺 How did you practice using ERL?

“I looked at online ERL cheat sheets before exams and saw questions where it was used in past papers. Apart from that, I have never practiced using ERL at home, because I didn’t know where to find questions like we did in class.”

A group of boys sitting at a table with a computer

Description automatically generated🡺 Did you find the experience of revising ERL easy or difficult?

“I didn’t spend much time deliberately practicing ERL, so it’s hard to describe it solely as easy or hard. It’s more of a thing where a random ERL question comes up whilst revising different topics, but I didn’t have a single set way to revise it – so I guess in that aspect it was kind of difficult as there’s not much option available.”

🡺 Would a translator for the ERL have aided revision?

A person in a suit using a computer

Description automatically generated“I be so for real; it would have genuinely helped as I didn’t spend any deliberate time on ERL at all. Even though I am confident in programming generally, so it wouldn’t have helped in terms of improving those skills but making me more familiar with the syntax and stuff would have just made the exam a bit easier”.

**🡺 Conclusions 🡸**

Tanish seems very comfortable with programming, so will make good assistance. He does reinforce that this solution would be very useful for all students, even himself as a comfortable programmer, which helps to bring focus onto the translator. It is meant to be designed as an educational tool over a powerful machine, so making it friendly in design and clear to use will be very important.

## computer science teacher

D

# translator research

This section will comprise of three different sections: researching current solutions for running ERL code, other translator designs, and some initial research into translator design. Afterwards I will then go into more depth about features – what my stakeholders want included and researching what I feel is important.

## pre-existing solutions for erl

### A white sheet of paper with black text Description automatically generatedTrace Tables

A white sheet of paper 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 in this different context it operates slightly differently.

A white sheet with black text

Description automatically generated🡺 Advantages:  
+ Already understood by the student so is familiar  
+ Helps to check the program’s logic is correct  
+ Used as a debugging tool to track the values and easily identify errors  
+ No software required, very accessible to all students

🡺 Disadvantages:  
- No syntax checking, so invalid inputs are treated as correct  
- The end result is based upon the individuals understanding of ERL, so if they misuse a feature it will be executed based upon that flawed understanding, with no proper feedback on whether it is acceptable  
- Extremely slow progress to create, especially with larger programs and with nested loops, so is too inefficient to use regularly, especially when repeated run-throughs are required due to errors.

🡺 Conclusions:  
Trace tables are too unreliable and slow to ever be used as a translator equivalent, however the positive aspects of simplicity are important to consider. Furthermore, the accessibility is important, and this highlights that as we need a translator that will be available to a wide range of students.

Finally, this could be considered as a potential implementation to the translator, to provide some form of debugging tool which the students are used to, without some of the complexities of more capable debuggers. If it is within the overall scope of the program, this could therefore be considered.

### manual translation to other languages

This is not a very technically proper approach but was a method I used in class to avoid creating a time-consuming trace table to run through my program. As I was comfortable with Python, after creating my ERL code, I would go line by line and enter the Python equivalent into a file, then run it to create my result, which would then allow me to check if the logic behind my program was correct.

🡺 Advantages:  
+ The computer produces a result at the end, therefore checking the logic behind the program  
+ Develops skills in ERL and a high-level language, providing the student with some real life skills  
+ Re-reading their own ERL is a form of checking for errors whilst translating to the other language

🡺 Disadvantages:  
- The ERL stage becomes very redundant, with not benefit to even including it when a high-level language can be used in the exam anyway, with no short-term benefit to writing the program out twice  
- Similarly to the trace tables, the student is the translator, so if their knowledge of the syntax or logic behind ERL is incorrect, then their translated program will not be accurate either, which defeats the purpose  
- Too time consuming, essentially re-writing the program is painful

🡺 Conclusions:  
Another poor current solution to translating ERL and should never really be used in a classroom. This highlights that the solution must be quick to use, as otherwise a student would never choose to use it. The link to real high-level languages could be considered, with a potential feature translating the ERL code to Python to link the two, however that may be too far out of the scope as is not the core feature.

### in conclusion

Overall, a translator for ERL seems very useful as there are currently no quick and easy solutions to the problem. In terms of focusing the project, it seems better to lean towards an Interpreter: the program gets executed on a click of a button rather than creating separate executable files. Whether the program approaches this line by line or the entirely at once is unimportant currently, as it will be abstracted to the user, but ensuring the program is easy and logical to use is critical.

## translators

Interpreters are usually designed around experienced programmers, which is not ideal for our solution, so despite these interpreters being very well made, I will be analysing them based on their use for new programmers – such as GCSE students.

### A computer screen shot of a code Description automatically generatednode.js in the terminal

This is a command line interpreter for JavaScript which can be called by “node” to launch the program.

A screen shot of a computer

Description automatically generated🡺 Advantages:  
+ Good error handling, with detailed messages pointing to the position of the error in the line – useful for beginners  
+ Multiple use modes including the shell (useful for a few lines), an editor (useful for questions), and a feature to run local files (useful for any larger projects) – making it good in a classroom.  
+ Has a basic .help feature which it tells the user about, providing access to the useful features which is needed.

A screenshot of a computer error

Description automatically generatedA screenshot of a computer

Description automatically generated🡺 Disadvantages:  
- No graphical user interface, making it difficult to use for new programmers who need something intuitive.  
- The terminal is often blocked on school devices  
- Complicated installation may not be feasible for students at home, and would require administrator access at school  
- Must memorise commands for fluent usage, tedious for beginners.

🡺 Conclusions:

Overall, I feel there are 3 main takeaways from node.

* Ensure that proper error handling is implemented, with good detail and pointing to the position.
* Make sure the interface is intuitive and is targeted towards beginner students.
* Prioritise accessibility, avoiding complex installs in the final product.

### Python idle

A screenshot of a phone

Description automatically generatedMy points will build upon the previous, expanding upon the features it includes.

A screenshot of a computer program

Description automatically generatedA screen shot of a computer

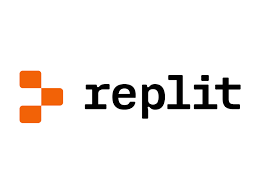
Description automatically generated🡺 Advantages:  
+ A vastly improved user experience, using a ribbon to implement the functionality, which is much more intuitive than commands for beginners.  
+ A much nice user interface, with syntax highlighting and proper indentation (which will also be important for ERL), which is much more pleasant than the command-line implementation.  
+ A greater array of features with “a debugger, with persistent breakpoints, stepping, and viewing of global and local namespaces” – which could be of use to a student

A screenshot of a computer

Description automatically generated🡺 Disadvantages:  
- As useful as some of these features are, they will never really be used by absolute beginners due to their complexity, so may be redundant.  
- An installation is still required, which may reduce accessibility  
- The official documentation is dense and hard to understand. There are still countless python tutorials on the internet – but in terms of an official equivalent for ERL – an easier solution would be required.  
- Having to save files in the editor before running can be annoying and complicated.

🡺 Conclusions:  
Visually Python IDLE should be a strong influence, as its focus on the code is important. In terms of improvement, just further decreasing the complexity would be good, such as by removing installation and adding more buttons to the project to make it even simpler.

### Replit

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

🡺 Advantages:  
+ Very accessible solution, and a website seems like the best solution so that it is available to all students with an internet connection without any install.  
+ The cloud servers ensure that all low-performance devices can still run code, therefore making it available to all students and very suitable for school devices  
+ When talking to a student who has used it before, Dylan Matthews, he said the main positive was the community aspect, with lots of users sharing their code and problems creating a learning environment.

A screenshot of a computer

Description automatically generated

🡺 Disadvantages:  
- Although there are some good aspects of the interface, such as the run button and separate editor and shell windows, overall, it feels very cluttered and overwhelming for a beginner.  
- Having account systems and being required to sign in is also overcomplicated, as ERL questions would only require short algorithms to be written, not requiring saving, and creating an account just slows the whole process.

🡺 Conclusions:  
As great as running the code on cloud servers would be for my project, this is beyond the scope of the project, and would be largely redundant due to the simple ERL code. However, a website seems like a promising idea, and I would therefore like to take this forward as it seems perfect for students at school.

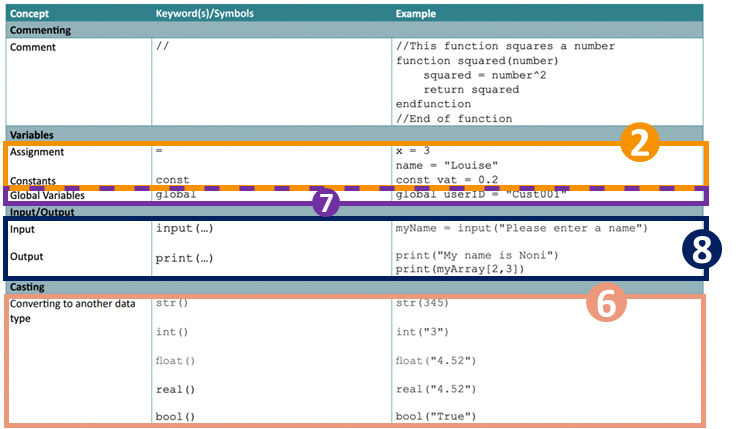
## Translator design

This is split into two sections: ERL specific and generally for translators

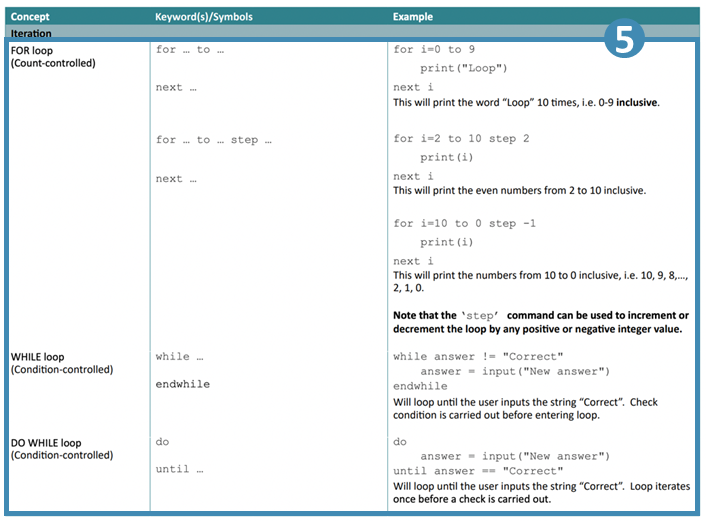
### For Exam reference language

It is important to recognise the entire set of instructions that need to be implemented, which are available on the exam specification at the following list:   
<https://www.ocr.org.uk/Images/558027-specification-gcse-computer-science-j277.pdf#page=27>

# Success critera

The main and obvious section of my criteria is the exam-board documentation of ERL.

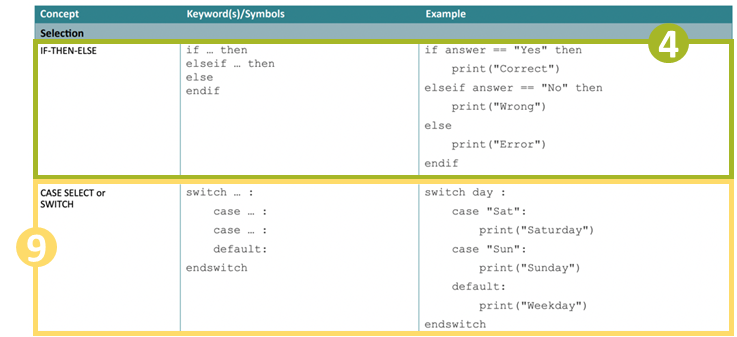
This has been divided into eleven modules which, when designing, I will go through, and iteratively design and develop and evaluate.

The contents of each module is very straightforward, and more detail is to follow, but highlighting the entire specification makes it clear that the entire codebase is going to be implemented.

These have been arranged in order of importance, and If less than 9 are completed, then the project will be deemed a failure as otherwise a lot of the key features will be excluded.

10 and 11 are also aims, but are more niche and less frequently used aspects of the ERL, so therefore could be excluded but it ideally should be involved.

Overall, the order is:

1. Arithmetic
2. Variables
3. Logic and Statements
4. If statements
5. Loops
6. Strings and casting

* **Mid-way Evaluation**

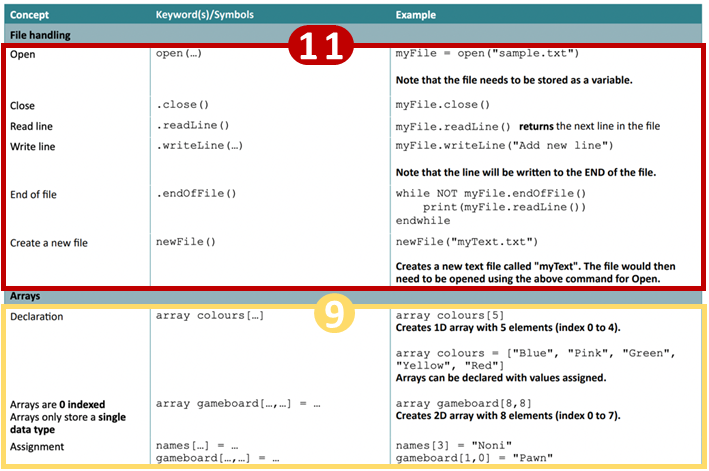
1. Functions and procedures
2. Input and Output
3. Switch and Arrays
4. String methods
5. File handling

* **Interpreter Evaluation**

1. User interface

* **Final Evaluation**

The final stage implementing the Interpreter into a website, which should be relatively straightforward as it is written in javascript but is important to provide accesibility for the user and to allow a proper means of use.





## 1: Arithmetic