Research Report for COS 333

Arnaud Zander Strydom
u23536013

February 25, 2025

1 Question 1

An esoteric programming language also known as esolang is a language created primarily as a joke(for fun), for artistic expression, code obfuscation,or as an experiment rather than for practical software development that developers are used to. These languages often have unconventional syntax, obscure logic, or extreme constraints for the porpusoe of deviating from conventional programming languages. The intention of esolang languages is not for these languages to be used in mainstream programming, although some of these languages have led to emphasis and conversations about some mainstream programming language features [3].

2 Question 2

2.1 a

Programming languages built on N-dimensional stacks, known as funges, allow for multidirectional execution rather than the conventional linear structure. It is a particularly unusual language because of its two-dimensional toroidal code space.

Advantage of Funges: Special flow control allows for innovative programming techniques by departing from models of linear execution.

Disadvantage of Funges: Compiling and debugging Funges is problematic because of its selfmodifying tendencid multi-directional PC [2].

2.2 b

Minimalist programming languages known as "Turing tarpits" strive for Turing completeness, which means that they can theoretically compute anything that can be computed using an arbitrary small number of linguistic elements. However, they are very unsuitable for real-world programming and significantly lack support for common operations.

Advantage: illustrates the effectiveness of minimalism: These languages demonstrate that Turing completeness can be attained even with a small set of instructions. Turing tarpits are still useful as an acedemic tool. It can be used pedagogically to demonstrate the fundamentals of programming languages and different facets of computability theory. They are also useful in algorithmic complexity theory, where binary combinatory logic and other similar systems have been used.

Disadvantage: Extremely difficult to use: Writing even simple programs requires excessive effort and is often unreadable [4].

3 Question 3

3.1 INTERCAL

An abbreviation for Compiler Language With No Pronounceable Acronym, was created in 1972, thus probably making it the first ever esoteric programming language. The inventors of INTERCAL Donald R. Woods and James M. Lyon invented it with the goal of creating a programing language that has zero similarities whatsoever to any other existing programing languages. The language almost completely suceeds in this regard except for the use of an assignment opperator. It is a one-dimensional language and uses binary. It was also the first language that had a wimpcode created for it due to its unuasual I/O requirements.

INTERCAL has a list of statements. Each statement is preceded by an optional label (a number in brackets), followed by DO, PLEASE, or PLEASE DO. The politesse of a program's statement identifiers is checked to make sure it stays within certain limits. DON'T, DO NOT, PLEASE DON'T, and PLEASE DO NOT can be used to write a statement that has no effect unless REINSTATED. If the statement is unlabeled and does not resemble an INTERCAL statement, it never has any effect at all.

Including the keyword Please in your code is required ,otherwise the compiler sees it as impolite and rejects the code. Having a balance is crucial because not using the Please keyword enough might be interpreted as rudeness, while an over use please might also lead to rejection.

Hello World in INTERCAL:

```
DO ,1 <- #13
PLEASE DO ,1 SUB #1 <- #238
DO ,1 SUB #2 <- #108
DO ,1 SUB #3 <- #112
DO ,1 SUB #4 <- #0
DO ,1 SUB #5 <- #64
DO ,1 SUB #6 <- #194
PLEASE DO ,1 SUB #7 <- #48
DO ,1 SUB #8 <- #26
DO ,1 SUB #9 <- #244
```

```
PLEASE DO ,1 SUB #10 <- #168
DO ,1 SUB #11 <- #24
DO ,1 SUB #12 <- #16
DO ,1 SUB #13 <- #162
PLEASE READ OUT ,1
PLEASE GIVE UP
```

[9]

3.2 BrainFuck

One of the most well-known esoteric programming languages is Brainfuck, which has influenced the development of numerous other languages.

Urban Müller created Brainfuck in 1993 in an effort to create a language that would allow him to create the smallest compiler feasible for the Amiga OS, version 2.0. He was able to create a compiler with 240 bytes. The 1024-byte compiler of FALSE served as the model for the language.

Imagine an infinitely long tape composed of cells, each one initialized to 0. There is also a movable data pointer which initially points to the first cell. There also two streams of bytes for input and output. Instructions are executed sequentially. The machine stops after executing the last one.

Brainfuck Commands:

- '¿' moves the data pointer to the next cell on the right.
- '¡' moves data pointer to the next cell on the left.
- '+' increments the value of the current cell.
- '-' decrements the value of current cell.
- '.' outputs the byte of a currently pointed cell in ASCII code.
- ',' reads one byte from stdin and store its value at the current cell.
- '[' if the current cell is 0 then jump to the matching ']'.
- ']' jump to the matching '['.

All characters other than '¿¡+-.,[]' are ignored.

Hello World in Brainfuck:

```
++++++++++  # set cell0 to 10
[           # loop until cell0 is 0
-           # decrease cell0
>           # move data pointer to the right (cell1)
+++++++     # increase cell1 by 7
<           # move data pointer to the left (cell0)
]
            # loop ends with cell1 set to 70 and data pointer on cell0
```

```
>            # move data pointer to the right (cell1)
++           # increase by 2
.            # print the result
```

[1]

4 Question 4

Contract based design is a software development methodology where a system is divided hierarchically into components and developed in a top-down way, using contracts as a means to divide responsibilities and manage the complexity of the system. It specifies the obligations and benefits of each component, ensuring that each part of the system operates correctly when used in conjunction with others.

Two languages that natively support Design by Contract are: Eiffel: One of the first languages to fully integrate DbC principles, where contracts can be explicitly defined within the language. Ada: Ada supports DbC via pragma directives that specify preconditions, postconditions, and invariants for procedures and packages [7].

5 Question 5

"Vibe programming," is a term that Andrej Karpathy came up with, the term refers to a software development approach where developers interact with AI systems using natural language to generate code, minimizing manual coding efforts. This method leverages advanced AI tools to handle the technical aspects of coding as well as the more menial tasks, allowing developers to focus on higher-level design and functionality [6].

Quality Control Challenges: AI-generated code may not always meet the desired standards or quality, leading to bugs or suboptimal performance. Without thorough review/debugging, these issues can persist unnoticed or cause undefined behaviour in the program [5].

Erosion of Fundamental Skills: Relying heavily or solely on AI for coding tasks might prevent developers from acquiring a deep understanding of programming principles, potentially diminishing their problem-solving abilities over time [8].

Security Vulnerabilities: AI tools might not always adhere to stringent security protocols, potentially introducing vulnerabilities into the codebase if not properly reviewed [5].

Loss of Creative Control: Giving coding tasks over to AI could lead to a declined sense of craftsmanship and creativity among developers, as they may become more focused on directing AI outputs rather than engaging in the coding process themselves [8].

References

[1] Radoslaw Bulat. "BRAINFUCK – LANGUAGE THAT WILL KILL YOUR BRAIN". In: *Esolang Journal* (2017). URL: https://thecodest.co/blog/brainfuck-language-that-will-kill-your-brain/.

[2] Alexios Chouchoulas. "Fungus: the Funge Machine". In: *Esolang Journal* (1993). URL: http://www.club.cc.cmu.edu/~ajo/docs/fungus.pdfe.

[3] Gyau Boahen Elvis. "Esoteric Languages: What are they, and why you should be concerned?" In: *Dev.to* (2024). URL: https://dev.to/gyauelvis/esoteric-languages-what-are-they-and-why-you-should-be-concerned-592d.

[4] esolangs.org. "Turing tarpit". In: *Esolang Journal* (1964). URL: https://esolangs.org/wiki/Turing_tarpit.

[5] Mehul Gupta. "What is Vibe Coding?" In: *Medium* (2025). URL: https://medium.com/data-science-in-your-pocket/what-is-vibe-coding-cf52c4efa867.

[6] Andrej Karpathy. "Vibe Programming: Redefining Software Development with AI". In: *Business Insider* (2025). URL: https://www.businessinsider.com/vibe-coding-ai-silicon-valley-andrej-karpathy-2025-2.

[7] Christian Lidström and Dilian Gurov. "Contract Based Embedded Software Design". In: *Theoretical Aspects of Software Engineering*. Ed. by Cristina David and Meng Sun. Cham: Springer Nature Switzerland, 2023, pp. 77–94. ISBN: 978-3-031-35257-7.

[8] Walse. "Vibe Coding: Redefining Creativity or Eroding the Soul of Programming?" In: *Dev.to* (2025). URL: https://dev.to/walse/vibe-coding-redefining-creativity-or-eroding-the-soul-of-programming-1on.

[9] Donald R. Woods and James M. Lyon. "THE INTERCAL PROGRAMMING LANGUAGE REVISED REFERENCE MANUAL". In: *Intercal Journal* (1973). URL: https://www.cs.virginia.edu/~asb/teaching/cs415-fall05/docs/intercal.pdf.