Summarize Section 2.3

This section starts out with some historical context of early computer programming. This section addresses the tolerance of slowness in interpretive systems from the late 1940s to the mid-1950s. The primary reason for this tolerance was the absence of floating-point hardware, requiring the simulation of floating-point operations in software, which was a time-consuming process. The announcement of the IBM 704 system with hardware support for both indexing and floating-point operations marked the end of the interpretive era, particularly in scientific computation. The text then delves into the debate over the credit for implementing the first compiled high-level language. While Fortran is often credited as the first, the question remains somewhat open. Alick E. Glennie's Auto code compiler for the Manchester Mark I computer is mentioned, but some argue that it was too low-level to be considered a compiled system. John Backus gives credit to Laning and Zierler at the Massachusetts Institute of Technology for their algebraic translation system implemented on the MIT Whirlwind computer. Despite these earlier works, the text highlights Fortran as the first widely accepted compiled high-level language. The subsequent subsections chronicle the important development of Fortran.

The next Section talks about the early development of Fortrans IBM 704 system and how in November the John Backus and his IBM group had drafted a report outlining the concept of Fortran, referred to as Fortran 0. The report expressed ambitious goals, aiming to combine the efficiency of hand-coded programs with the ease of interpretive pseudocode systems, and even proposed eliminating coding errors and debugging. Fortran was developed in an environment where computers had limited memory, were slow, and somewhat unreliable. The primary use of computers was for scientific computations, and there were no efficient programming methods available.

This section talks about the Fortran 1 Overview. Fortran I's control statements were based on IBM 704 instructions, and it's unclear whether the 704 designers influenced Fortran I's control statement design or vice versa. Notably, Fortran I lacked data-typing statements. Variables starting with I, J, K, L, M, and N were implicitly integers, while others were implicitly floating-point. This convention reflected the common use of these letters as subscripts in scientific notation. During Fortran's design, the development group made an audacious claim that the machine code produced by the compiler would be only about half as efficient as hand-coded equivalents. Despite initial skepticism, the Fortran team achieved remarkable efficiency through optimization efforts, with a significant portion of the 18 worker years dedicated to this task. Fortran's early success was evident in an April 1958 survey, where roughly half of the code for IBM 704s was already being written in Fortran.

Fortran, a historically significant programming language, underwent several versions and improvements over the years. Fortran III Developed but not widely distributed. Fortran IV 1960-1962 Became one of the most widely used languages. Standardized as Fortran 66. Improvements over Fortran II included explicit type declarations for variables, a logical If construct, and the ability to pass subprograms as parameters. Fortran 77 1978 Replaced Fortran IV as the new standard. Retained most Fortran IV features and introduced character string handling, logical loop control statements, and an If with an optional Else clause. Fortran 90 1992 Dramatically different from Fortran 77. Major additions included dynamic arrays, records, pointers, a multiple selection statement, and modules. Fortran 90 allowed recursive calls to subprograms. Introduced the concept of removing certain language features from earlier versions, promoting evolution. Fortran 95 1997 Continued the language's evolution with only a few changes. Introduced the Forall iteration construct to facilitate parallelizing Fortran programs. Fortran 2003 Added advanced features such as support for object-oriented programming, parameterized derived types, procedure pointers, and interoperability with the C. Fortran 2008 The latest version introduced support for blocks to define local scopes, co-arrays for a parallel execution model.

Question 3

When choosing Between Fortran 95 and C++ or Java I would want to choose for my scientist something they know, and I highly doubt any of my scientists no Fortran I feel that it might make it a hard learning curve. But I do feel that if the scientist needs something in numerical and or scientific computing, I would choose Fortran but if they need anything else developed I would choose the other languages.

Question 4

The limitations of the existing language, FLPL (Fortran List Processing Language), prompted McCarthy to envision a new language that could fulfill these requirements. Upon returning to MIT in fall 1958, McCarthy and Marvin Minsky established the MIT AI Project with funding from the Research Laboratory for Electronics. Their initial focus was on creating a software system for list processing, intended for the implementation of McCarthy's proposed program, the Advice Taker. This initiative laid the foundation for the development of LISP.

Lisp's code and data share a uniform structure (S-expressions), making it highly flexible and allowing programs to be easily manipulated as data. The ability to treat code as data enables powerful metaprogramming and facilitates the development of sophisticated AI algorithms. Lisp's dynamic typing allows for more flexible and expressive programming, facilitating rapid prototyping and experimentation. Lisp's interpreted nature allows for interactive development, providing a REPL (Read-Eval-Print Loop) where developers can experiment and evaluate code snippets in real-time. Lisp's symbolic expressions (S-expressions) allow for easy representation of symbolic knowledge, making it well-suited for symbolic reasoning and manipulation. Lisp excels at symbolic computation, which is fundamental to many AI tasks, such as expert systems, natural language processing, and knowledge representation.