1.This book talks about the latest research and ideas in Computational Thinking Education (CTE). It includes work from different educators and researchers, especially focusing on how to teach CTE to young students (K–12). The idea is based on the legacy of Seymour Papert, who believed in using computers to help children think and learn better.

CTE became important as computers started becoming more common in everyday life. Early programming efforts started in the 1950s and 60s, like the BASIC language created at Dartmouth College, which helped students learn to code.

Another key development was the Logo programming language, developed by Papert and others in the 1960s. It was designed to help children learn through

building and exploring ideas. Papert believed that computers could help all children become confident .Despite increasing interest, there's still confusion about what CT really means. Different approaches to teaching CT in schools focus on basic computer literacy, learning specific programming languages, or using computing tools to complete tasks. However, experts still lack a clear, unified definition of CT. The NRC report emphasizes the importance of CT in today's tech–driven world and the need to address educational and policy challenges around its implementation.

2.introduction to computational thinking education by siu–cheung kong, Harlod Abelsob and ming lai

3.Despite the growing interest in Computational Thinking (CT) in K–12 education, there is still confusion and lack

of clarity about what CT truly means and how it should be taught. While CT is increasingly important in the modern, technology–driven world, many people still mistake it for just programming. Even experts and researchers do not fully agree on a clear definition of CT. This lack of consensus makes it difficult to implement CT effectively in schools, even though its importance for students' future learning and problem–solving is widely recognized

4. i) To introduce computational thinking
ii) To identify the specificity of computation thinking
iii) To measure students computational thinking
iv)To illustrate how computational thinking and programing education can be implemented.

5.

1. Lack of a Clear, Widely-Accepted

Definition of CT

Although there is growing interest in Computational Thinking, researchers and educators still struggle to agree on exactly what CT means.

The introduction notes that even expert groups (like the 2009 National Academies workshop) could not fully agree on one precise definition of CT.

Gap: There is no single, clear, standardized definition of Computational Thinking used consistently across research and practice.

2. Confusion Between CT and Programming

Many implementations treat CT as just learning programming languages or basic computer use. For example:

- Using computers to create documents
- Learning languages like Java or C++

But the authors argue that CT should be deeper than programming involving reasoning, problem solving, abstraction, decomposition, and algorithmic thinking.

3. Limited Focus on Fundamental CT Concepts in Curriculum
The introduction notes that many efforts to integrate computing in K–12 focus on generic computer tasks (like typing or making slides) rather than core CT skills.

Gap: Existing curricula and educational tools do not consistently teach the foundational CT ideassuch as decomposition, pattern recognition, abstraction, and algorithm design.

4. Inconsistent Emphasis on CT Across Schools
Even with increased interest, CT adoption in institutions is still uneven — some

schools teach basics, while others barely address it.

Gap: There is no consistent or systematic integration of CT into school curricula globally.

5. Need for Better Understanding of How CT Should Be Taught

Although many programs exist, there's limited research on effective teaching methods that genuinely develop CT thinking in learners.

Gap: Research on how to teach CT effectively — beyond tools and technologies — is still underdeveloped.