Computational Thinking the Intellectual Thinking for the 21st century.

J. Shailaja
Faculty of Computer Studies,
National Academy For Learning
Bangalore, Karnataka, India
shaila_jm@rediff.com

Dr. R. Sridaran, Dean
Faculty of Computer Applications,
Marwadi Education Foundation’s Group of Institutions,
Rajkot, Gujarat, India
sridaran.rajagopal@gmail.com

Abstract
Computational Thinking (CT) and its implementation in the K-12 level is the burning need for the 21st century. In order to be successful in a world that is increasingly dependent on technology CT is essential. Fundamental skills acquired by young student’s plays a very important role in shaping their career, so nurture every child with CT precisely at K-12 levels. This paper explain what, why and mainly how to teach CT at different K-12 levels. Blended learning will be utmost appropriate to teach CT in the existing situation. As the school curriculum is already packed, besides inadequate hardware and internet support. We can bring about a generation of computational thinkers and digital creators by prompting the approach as specified in this paper.

Keywords – Computational Thinking; blended learning; generalisation; algorithmic thinking; abstraction.

I. INTRODUCTION
Computers are undoubtedly the best invention of the millennium. Computers have swamped the world wherein any field or subject culminates into functioning in a facade of a computer. Through Computational thinking (CT) one can excel in life as they are better equipped to solve glitches. CT can be imparted, and the earlier we start the operative it will be. CT is a focused approach to problem solving, incorporating thought processes that utilize abstraction, decomposition, visualisation, pattern recognition, algorithmic thinking, evaluation, and generalisations [18]. Guileless description for CT can be an intellectual thinking which is a step ahead of common sense, and is unique in humans. People with CT skills can think algorithmically, visualise stuff with abstraction, compare and contrast situations to arrive at a solution by decomposition.

Computational Thinking is not necessarily a pathway to producing software or hardware artefacts. But to solve problems in diverse contexts of everyday life – even in the absence of computational devices. Some aspects of CT have been used even before the existence of modern computers. The original computing method was counting by human fingers, centuries ago. In this sense, CT may be a blend of thinking elements previously related to other thinking paradigms, such as logical, math, analytical or engineering-oriented thinking. Hence, undertaking computing should be a way of enhancing CT skills [10]. This research is diverse.
CT INPUT PROCESS OUTPUT

<table>
<thead>
<tr>
<th>CT T</th>
<th>INPUT</th>
<th>PROCESS</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>![Input]</td>
<td>![Process]</td>
<td>![Output]</td>
</tr>
<tr>
<td>Problem</td>
<td>2 + 3</td>
<td>![Process]</td>
<td>5</td>
</tr>
<tr>
<td>Human</td>
<td>![Input]</td>
<td>![Process]</td>
<td>![Output]</td>
</tr>
</tbody>
</table>

Fig. 1 IPO cycle in a computer is similar to human processing. Computational thinking is human thinking.

A cost-effective infrastructure for building ubiquitous collaborative learning spaces to build a learner-centric, service-based architecture to transform existing traditional learning spaces into intelligent ambient learning environments is the challenge. This can be proficient with CT. Effective real-time collaboration and resource sharing between learners and their instructors, which facilitate team-based, collaborative, and inquiry-based approaches towards constructing CT [3]. Computing devices and online platforms for coding like Scratch, Codeacademy, and unplugged activities can be used. With the integration of CT, students will reason abstractly and quantitatively, look for structure and regularity. Coding is one of the new literacies, a new way of thinking and critically solving problems in this digital age, and a definite method to relate deep intellectual CT [2]. Teaching algorithm is the simplest and suitable for all age groups. Take real-time examples and motivate the students to think algorithmically for any given job. This can inspire the students as the complex, smart and impossible looking gadgets functioning can be understood. It can conduit the gender prejudice too.

Ancient schooling was all about real life instances in the form of stories. At present we teach students the concepts but fail to instil their applications, later kids also don’t relate to it. They will discern the concept but fail to apply them. A meek example: formula to calculate the area and perimeter of a square/rectangle is known to a child age 10 and above. But fail to calculate the cost to fence the garden or carpet the room. Concepts are known-but it is not of any use if they fail to apply them when required. Barcelos et al. [10] concludes that the transition from the mathematical representation of the sequence to its algorithmic representation is not straightforward for most of the students. Besides, it is shown that the root causes for this difficulty rely mainly on the differences between both semiotic representations. It can be conjectured whether such students would present the same difficulties if they had been first exposed to Computational Thinking foundations in early years, instead of the so-called “traditional” approach. In the traditional approach there will be few exceptions in a class who can arrive at an answer, they are the gifted once with computational thinking. So why not gift every child with this amazing skill, it can be imparted.

A gradual shift from face-to-face to online learning is an unavoidable force of change as students are digital natives and need immediate data, so is the work force. [8] To stay ahead of the tide all students, institutions and instructors should shift towards online teaching. Self-study, problem solving, abstraction, decomposition and generalisation are the key elements to succeed in an online environment, and these skills are in turn computational thinking’s core elements.

The learning environment in a typical classroom can be characterized as active interactions between learners and instructors or between learners and other learners. Teacher control model blended learning environment is our solution to extend the school learning time and space to anytime[4]. This can be convenient as adding another subject to the already packed school curriculum can be very demanding. So with blended learning this can be achieved partially at home with adequate web-support. The aim is to choose a mix that will highly motivate the students, and assist them in successfully mastering the course [5]. Technology Enhanced Learning (TEL) is described based on the context-can be any information that can be used to characterize the situation of an entity [6]. There has always been a gender gap in technology, given a little extra time and support girls can outperform boys. This can be accomplished in blended learning as girls get that bonus time to work at their pace, also starting in early at K-12 levels builds the confidence. Abstraction is an important aspect of computational thinking this can be attained by finding out context acquisition, representation using proper Human Computer Interface (HCI), evaluation techniques to data sharing and privacy with interaction. Computational Thinking can come in handy to manoeuvre any kind of HCI.

Cooper et al. says one fundamental idea behind computational thinking is to break a problem up into its parts, abstracting out details so that patterns can be more easily seen. An algorithm is then constructed to automate the task of solving the problem itself. The point of CT is not to give a precise number of steps to attempt in a particular sequence, but rather to train one’s mind to recognize how a current issue might be similar to a problem that already has a known solution [22]. Even though the idea of “Computational Thinking” is enjoying a healthy buzz in the K-12 space, there is as yet no cohesive or well agreed-upon theory of what it looks like in practice and how it can be assessed and measured in generic and generative ways. [20]
II. RELATED WORK
Computational thinking is still in its infancy so one can find a lot of survey papers presenting the need to bring in CT into K-12 settings. However implementing it, a solid curriculum to enforce it, is still hanging. Few researches to find the explicit pattern of CT are explained. An effort by Basawapatna et al. [1] to identify if students can apply the knowledge obtained from programming games to creating science simulations. There has been less research, in general, on the actual knowledge/skills students acquire.
CT with creative thinking was clubbed for college students with non-computational background to understand concepts. To unlock CT exercise like – choose an everyday object and describe all its characteristic and objective, writing stories individually and later try to connect it and make it bug free as well. To learn about abstraction and function characterization by identifying properties of an everyday object [13]. ACM, CSTA and CSA have come up with curriculum to teach CT in K-12 level, but they only deal with why and what to teach. How to teach is missing in their content. So it has not been engaging and implement well. Only a joint work between professionals of these two areas (Theoretical Computer Science and Education) will result in the successfully introduction of CT at school [19]. Teaching computing for 10 years with a proclivity towards research in CT here are a few easy but effective angles to implement CT. This paper gives a detailed description of what to teach, how to teach and its effective application in any K-12 setup.

III. IMPLEMENTATION AND EVALUATION
Information Communication and Technology (ICT) is termed as Computer education in school and is mostly used to teach other subjects and many e-learning materials explain only how to use it. ICT deals with what and how, whereas CT deals with why. Computing is better as technology changes quickly. By the time the students are out of school the software learnt is outdated. For instance a student is proficient to work on MS Word 2003, know exactly which option to choose and the steps to accomplish it. When updated to Office 2007/2010 the student has to unlearn stuff he/she has learnt and will keep looking at all the ribbons to find the option. Students know only how to use a computer and this is not CT. In contrast a child with CT skills will think critically and reason out the option by generalising it and choose the right ribbon to go on the option. Educators must understand better how to teach computing so that students have the knowledge to successfully apply CT skills in their fields.

CT skills were appreciated as skills only for computer scientist then moved down to higher education. With some earnest efforts, CT has reached K-12 age groups which is practically possible to be implemented without access to computers. CT skill is engaging students in doing computing, not as a tool, but as a problem-solving way of reasoning [10]. CT concepts must appear as early as the primary grades, and then continue through the secondary grades and beyond [27]. One way to do this is to incorporate CT modules into core education courses to expose future teachers to this idea. “If we wanted to ensure a common and solid basis of understanding and applying CT for all, then this learning should best be done in the early years of childhood”[23].

There is always a mismatch between students learning and the industry requirements. As the problems they face are new which cannot be built with logic rather requires abstraction, critical thinking and problem solving. Which are the important aspect of CT and very few possess it. This can be accomplished by the formation and evaluation of the activities specified below. CT is different from computer science, it is not word processor, spreadsheet etc. and not even programming- it is a skill used by computer scientists which can be acquired by learning computers. As CT does not need a computer and just goes on in the head. It is a mental tool.
ACM/IEEE came up with a model for college computer science major programs. This model identifies the following “core” subjects in 13 distinct areas that should cover. The same list can be implemented into grades 9-12 partially by abridging it yet teaching the concepts through CT. Table 1.
<table>
<thead>
<tr>
<th>Topics</th>
<th>Concepts which can be implemented to enforce CT into grades 9-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Simple step-by-step instruction from real world like making juice to explain the working of any embedded system like an AC, washing machine.</td>
</tr>
<tr>
<td>Architecture</td>
<td>Binary and Hexadecimal numbers, conversion from binary to decimal and vice versa, Von-Neumann architecture.</td>
</tr>
<tr>
<td>Discrete structure</td>
<td>Functions- using spreadsheet. Sets and relations – using database. Logic- simple programs like patterns and game development.</td>
</tr>
<tr>
<td>HCI</td>
<td>Brainstorming sessions to better a badly designed website or app.</td>
</tr>
<tr>
<td>Information management</td>
<td>To create tables using database software for a school or library. Building relation between tables, query and generate reports.</td>
</tr>
<tr>
<td>Intelligent system</td>
<td>Explain the working of the following: Expert system- Medical diagnosis, mineral extracts.</td>
</tr>
<tr>
<td>Network centric computing</td>
<td>Online booking of airways, railways, movie tickets. Client server computing- uploading and downloading files.</td>
</tr>
<tr>
<td>Operating system</td>
<td>Device management- driver installation, handshaking, check digit.</td>
</tr>
<tr>
<td>Programming fundamentals</td>
<td>Problem solving- area, perimeter of a square/rectangle, speed, simple interest.</td>
</tr>
<tr>
<td>Social and professional issues</td>
<td>How to be netiquette using case studies. Risks- hacking, virus, spyware, phishing, pharming.</td>
</tr>
<tr>
<td>Graphics and visual computing</td>
<td>Difference between the following: Bitmap and vector images, using paint and clipart.</td>
</tr>
</tbody>
</table>
It is found that the students in non-CS disciplines come from diverse backgrounds and many are likely to have limited understanding of computing concepts that are used as the basis for CT. Without a basic level of understanding, these students will have a very difficult time developing CT fluency. As noted by Epstein et al. [14] and Shell et al. [15], one cannot think beyond the boundaries of one’s cognitive tools or knowledge. Students who are forced to use CT based on concepts that they do not understand may become frustrated, potentially lowering both their motivation and their self-efficacy. As both student motivation and self-efficacy are strongly connected to student learning [16][17], To tackle this problem CT has to be embraced at least at higher secondary levels by schooling as given in the table above.

Counting and arithmetic are taught in preparation for more complicated mathematics education in the future, finger painting and colour mixtures introduce art, and the alphabet is a precursor to reading and writing. Similarly, CT is a particularly elegant way to introduce computer science at the K-8 grade level, due in large part to the fact that computers are not actually required. In much the same way as math is practiced with pencil and paper before a calculator is introduced, thinking computationally can be practiced through stories and activities well before a machine is necessary [28].

Mother of CT, Wing’s questions are:
What are effective ways of learning (teaching)CT by (to) children?
- What concepts can student’s best learn when? What should we teach when?
- What is our analogy to numbers in K, algebra in 7, and calculus in 12? [25]
Fig. 2 Answers these questions.

1. Grade K-2 levels
Traces of CT in education already exist, but it needs to be nurtured further.
Right from the Montessori level that is K-2 levels kids learn by doing (hands on) resembling:
- Writing alphabets on sand or running their hands on wood or solid blocks similar to working on a computer.
- Algorithmic thinking is learnt in buttoning a shirt or beading a string with beads in ascending or descending order.
- Take a tour of the school and remember where each room is and give directions by visualising.
- Blind fold the teacher and students give instructions to do a specific job or reach a place.
- Going over a rhyme again and again, or retrace on alphabet to attain perfection (recursion/iteration
- In paint or Tux paint encourage the kid to draw the growing of a plant, water cycle, growing of a butterfly, volcano eruption and show working of any device (decomposition)
- Draw 3 D looking objects on a 2 D surface; give the illusion of moving things. E.g. add smoke to show a car moving.

The 3 CT skills which can be engaged at these levels mainly are Visualization- V, Pattern recognition- PR and Generalisation – G with other skills.
As per the ACM curriculum for K-2 levels what to teach along with how and the CT skills attained.
- Input and Output device:
  Input-To put in information like humans –eyes, ears same as keyboard and mouse. (Fig 1)
  Output -To take out information: tongue, skin similar to monitor, printer. (G)
- Work independently – on software’s like paint, tux paint, notepad, WordPad. (PR)
- Terminology – know the appropriate use of terms. (G)
- Multimedia- use an educational CD. (V)
- Social and ethical behaviours – rules to follow in the lab; use the computer only under an adult’s supervision. Safety and security from a technical perspective. (G)
- Work collaboratively – draw a picture label it, make six difference, design a maze (logical thinking)
- Use technology resources – give pictures and make a story or type a story in a word processor and draw a picture using drawing software and paste it. Write the steps to reach the computer lab. Steps to switch on and turn off the computer. (algorithm)
• Concept of 1’s and 0’s – computers only work with it as it is on or off like a regular fan or a toggle switch. (V)
• Sorting and searching – to stand in alphabetic order then according to height. Search for a particular book in a stack of books. (PR)

In addition to the ACM list:
- What are machines, types of machine and why machines for repetitive tasks, speed and accuracy. (G)
- How to make juice, sandwich …one can give instructions and another should execute it (algorithmic).

2. Grade 3-5 levels

The CT skills which can be promising at grades 3-5 are Decomposition-D, Algorithmic Thinking-AL and Evaluation-E.

Algorithm is the most powerful word in Computing. It is the heart of computer science and the part to focus on with children because once they grasp it, everything else is easy. (Fig. 3)

As per the ACM curriculum for 3-5:
• Keyboard – Tasks to type by integrating it with other subjects like a science topic as how a plant grows or how heart beats, digestion process in steps. Equally a social science topic such as how a civilization developed. (D, AL)
• Technology in daily life – why a specific gadget is used and places were a machine is used and why. (E)
• Personal consequences – with case study, quiz, story to bring in awareness, privacy, dos and don’ts on the internet. (E)
• LOGO/Scratch – learn angles, execution of a command, repeat command to learn the concepts of loop, arithmetic calculations, logic building by making patterns, simulate robotic movements. (D, AL)

Way back in 1970 Papert’s LOGO can be used very effectively to teach angles and math concepts till date. Robotics is also on the same lines. Drawing polygons using repeat command teaches geometry and shape. Repeat command to draw a circle explains why machines are used for repetitive jobs. Kids make wonderful patterns on LOGO which is beyond the imaginations of adults. This involves a lot of logic and analytical reasoning.

In addition to ACM curriculum:
• CT is not teaching to work on Word, PowerPoint or Excel. However the same can to customise to cultivate CT skills.
• Word processor – basic features of the software should be taught and encourage them to build a simple board game and type the instructions. This can be printed and played with another player by following the instructions. If the instructions are clear the game can go efficiently or else it will be a chaos. Make the students understand the need for step by step instructions and its mandate. (AT)
• Concept of algorithms and working of any application can be bought in while teaching the students to save a file. Difference between save and save as a different procedure is called. Concept of primary, secondary memory, buffer and clipboard can be elucidated here. (D, E)

Δ May be the brain also uses a complex algorithm, which we are yet to discover.

Fig. 3 Algorithmic Thinking

- Presentation software – make a flip book with automatic transition to realise how animation is done.
- deadlock situation can be unveiled with limited stationary in a group and explain pipelining for effective use of resources.
- Jumble the steps to do a task like how to save a file and reorganise it.
- Finding similarity in everyday objects and how it has been improvised. For example: Type writer – keyboard- dot matrix printer

Δ A quill and ink to an ink pen then an inkjet printer
Carbon sheet- Xerox machine – laser printer.
3. Grade 6-8 levels

The CT skills which can be favourable at grades 6-8 are abstraction – A, Critical Thinking- CrT, Computing - C

To teach CT programming language such as C++, Java, Python are not suitable for K-8 students. It only increases the anxiety and fear in a child. This can mislead the students in not choosing any course related to Computers. Hence enjoyable, interactive and visual programming language like Scratch, Alice can be an appropriate substitute. Nonetheless these languages need extensive internet support, which is not feasible, is most schools. Basic, Visual Basic (VB) can be decent alternative.

ACM for 6 to 8

- Trouble shooting – thinking logically to discern what could have gone wrong by means of backtracking (Cr T)
- Changes in technology – how and why these changes are effecting office and all places. (A)
- Exhibit legal and ethical behaviour – concept of IP address, routing and not being anonymous. (A)
- Software tools – Basic, VB by doing simple programs to appreciate what programming, Flash to know what goes into animation, Photoshop to get acquainted with the techniques (C)
- Multimedia tools – to create a movie or animation to understand graphic, audio and video editing. Concept of a theme, storyboard, characters…. To make a motion picture. (A)
- Hardware – Know the working of the hardware parts and build relations with other device like typewriter to keyboard to dot matrix printer. (Cr T)
- Sets and graphs- concept of a pixel, how to represent it, making a screen saver. (A)
- Introduce if statement by using real time circumstances similar to, decide on the age to vote, check for user name and password. Multiple and nested if to decide on the grade. (C)
- Loops for repetitive jobs with the same accuracy. Types of loops entry controlled and exit controlled. (C)

In addition

- Differentiate input and output device and their explicit use. (Cr T)
- Writing algorithms for real embedded machines. (Cr T)
- Types of networks, topology and its routines. Online transactions its advantages and disadvantages. (A)
- Difference between humans and machines- making a list of things which a computer cannot do. (Cr T)
- Design flyers using publisher/word processor – for planning, layout design and use of templates.-generalisation and decomposition
- Differentiate between bitmap and vector images with jpeg and clipart graphics- (A)
- Spreadsheet – formulas and functions for calculations with cell reference. -Conditional formatting – design a Sudoku,Crossword, area calculator.- If analysis – personality quiz, conditional branching, decision making. (Cr T, AT)
- Database –Creating tables for school, library management. Sequential and random access of records. Verification and validations for data types can be explained well. (Cr T, AT)
- Installation of hardware or software explaining itsrequirements. Requisite for a driver to be installed. Once installed how it carry on with the drudgery. (A)

4. Grades 9 to 12

We need to broaden our array of computing activities. While the shift towards designing authentic applications is an important step into the right direction, we also need to realize that designing games is not the only acceptable application to achieve this goal[21]. Now we identifyCT is important educationally. It requires logical thinking and precision. It encourages innovation, collaboration, and resourcefulness: pupils apply underlying principles to understand real-world systems, and to create purposeful artefacts. This combination of principles, practice, and invention makes CT an intensely creative subject, suffused with excitement, both visceral and intellectual. More broadly, it provides a “lens” through which to understand both natural and artificial systems. This is a perfect age to undertake all CT skills. The cases should be designed carefully by teacher before class. A successful case must be not only related to the knowledgepoint which will be taught in class but also being familiar to students so that they are absorbed to it. [11]

To teach CT programming language such as C++, Java, Python, VB are suitable for grades 9-12.Not every child needs to learn a programming language, but without some understanding of how code works and how it affects our lives, we may be depriving young people of new avenues to creativity, and valuable skills for the job market.[29]

- Simple games like slot machine can be developed in VB.
- Artless programs like finding area and perimeter of a geometric shape, solve a quadratic equation, finding the speed, Simple and compound interest calculator.
- Concept of conditional branching, loops, counter and accumulator.
- Make a screen saver in VB using random function for circles to come in different size and location.
- Mouse click event can be used to develop a simple game where objects appear at random and when clicked an accumulator to calculate the points. To teach concepts of conditional branching and iterations.Search and sort later.
• Writing code to generate number patterns, cryptography and strings.

Programming languages will come and go, but technology will stay. Everyone should at least try programming. Teach what they want to learn and what interest’s students and establish the concepts of CT in the whole process. Moreover, citizens able to think in computational terms would be able to understand and rationally argue about issues involving computation, such as software patents, identity theft, genetic engineering, electronic voting systems for elections, and so on. In a world suffused by computation, every school-leaver should have an understanding of computing [24].

European qualifications framework (EQF) is a software suite that allows educators to embed the EQF into their course designs and educational practice of online and blended teaching. This model defines the strategies for bridging gap between the prerequisites and the learning outcomes as a set of rules [7]. This Learning analytics can be used to gauge the CT effectiveness. The interest and number of students willing to take up computing and its branches has increased two folds after this implementation. Join hands to create future computational thinkers.

CONCLUSION

There is an immediate need to teach CT as early as elementary level. Once students clench to think computationally, students do perform well as they can relate, understand concepts better. Students can analyse, evaluate and solve problems better in school, work and life in general. So CT is a skill for a better life. We want our children to understand and play an active role in the digital world that surrounds them, not to be passive consumers of an opaque and mysterious technology. Asound understanding of computing concepts will help them see how to get the best from the systems they use, and how to solve problems when things go wrong. Companies across all fields are hungry for computational thinkers as it is such a 21st century intellectual skill.

REFERENCE


[21] Yasmin B. Kafai, Quinn Burke, “The Social Turn in K-12 Programming: Moving from Computational Thinking to Computational Participation”, SIGCSE ’13, ACM.


