Atomic Energy Education Society, Mumbai

Class XI **Computer Science**

BASICS OF COMPUTATIONAL THINKING **Module 01/02**

Introduction

Computers can be used to help us solve problems. However, before a problem can be tackled, the problem itself and the ways in which it could be solved need to be understood.

Computational thinking allows us to do this.

Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions. We can then present these solutions in a way that a computer, a human, or both, can understand.

Computational Thinking (CT) is a problem solving process that includes a number of characteristics and dispositions. Computational thinking involves a number of **skills**, including: Formulating problems in a way that enables us to use a computer and other tools to help solve them. Logically organizing and analyzing data.

Computational Thinking is essential to the development of computer applications, but it can also be used to support problem solving across all disciplines, including the humanities, math, and science. Students who learn CT across the curriculum can begin to see a relationship between academic subjects, as well as between life inside and outside of the classroom.

Simple Daily Life Examples

- Looking up a name in an alphabetically sorted list
  - Linear: start at the top
  - Binary search: start in the middle
- Cooking a gourmet meal
  - Parallel processing: You don't want the meat to get cold while you're cooking the vegetables.
- Cleaning out your garage
  - Keeping only what you need vs. throwing out stuff when you run out of space.
- Doing laundry
  - Pipelining the wash, dry, and iron stages.
- getting food at a buffet
  - Plates, salad, entrée, dessert stations

The Principles of Computational Thinking:

These are key techniques that will help you think computationally through a complex problem (challenge, or task) before writing a single line of code.

1. Decomposition
2. Pattern Recognition
3. Abstraction.
4. Algorithm Design

Decomposition

This is breaking down a complex problem or system into smaller, more easily solved parts. These smaller problems are solved one after another until the bigger complex problem is solved.
i.e. Breaking down data, processes, or problems into smaller, manageable parts is termed as decomposition or modularization.
Decomposition allows us to break a complex problem into a number of smaller and independent modules which facilitates us to work as team where every team works to solve a smaller and independent module and later we collaborate all such efforts to solve the complex problem.
Decomposition allows us to divide and conquer complex problems. It also allows us to localize faults as and when it occurs and subsequent quick rectification.
Decomposition introduces the concept of layered approach where a complex problem is decomposed into a number of independent and loosely coupled modules. This modular approach helps us to incorporate modifications at a later stage easily.

Pattern Recognition

Observing patterns, trends and regularities in data.Looking for similarities among and within problems.This is done to find and establish a generalized approach which can be used to solve similar other problems.

Generalization

Generalizationis the tendency to respond in the same way to different but similar problems.
For **example**,
A group of statements used to find sum of a finite number of integers can also be used to find product of such integers. i.e a for loop which can find sum of first 10 integers can also be modified to find average or product of such integers.
Thus through generalization we devise methods and algorithms to solve problems similar to the problem at hand.