Department of

Computer Science and Engineering

Title: Introduction to computational thinking

Computational Thinking and Problem Solving

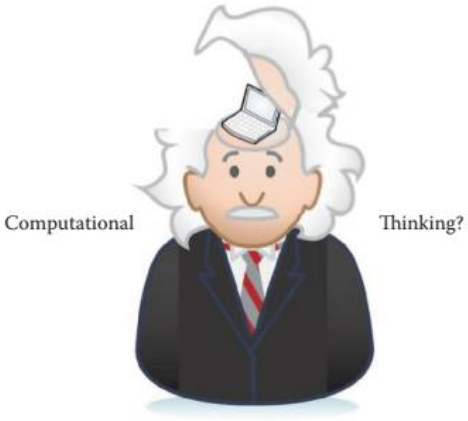
CSE 100

Green University of Bangladesh

1. Objectives:

- Define computational thinking.
- Describe the four pillars of computational thinking and their benefits
- Apply the pillars of computational thinking to solving a problem

2. Computational Thinking:



Computational thinking is an approach to solving problems using concepts and ideas from computer science, and expressing solutions to those problems so that they can be run on a computer. As computing becomes more and more prevalent in all aspects of modern society -- not just in software development and engineering, but in business, the humanities, and even everyday life -- understanding how to use computational thinking to solve real-world problems is a key skill in the 21st century. Computational thinking involves breaking down a problem in smaller parts, looking for patterns in those subproblems, figuring out what information is needed, and developing a step-by-step solution. Computational thinking is built on four pillars: **decomposition, pattern recognition, data representation and abstraction, and algorithms**. This session will introduce you to the four pillars of computational thinking and shows how they can be applied as part of the problem-solving process.

Daily Examples of Computational Thinking:

- Looking up a name in an alphabetically sorted list (Binary Search)
  - ✓ e.g., 100 names per page in list of 150,000 names
  - ✓ How to minimize the number of pages to look at?

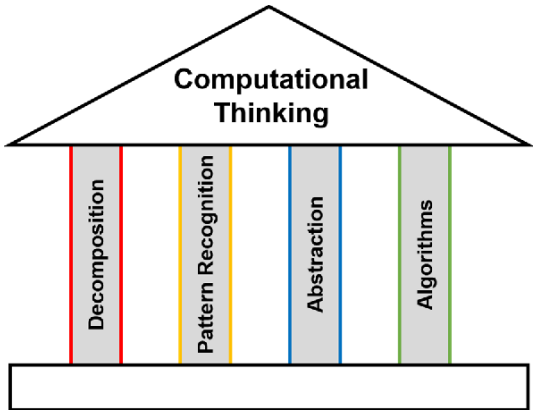
- You and your friend are buying tickets for a movie (Parallel Processing).
  - ✓ There are three independent lines
  - ✓ How do you get your tickets ASAP?

- What is the best way to serve 20 pizzas to 60 hungry students? (i.e., How do you minimize the time for everyone to get pizza?)
  - ✓ One table with all pizzas (the usual case) – NO!
  - ✓ Five tables with four pizzas each

In a nutshell, Computational thinking is ---

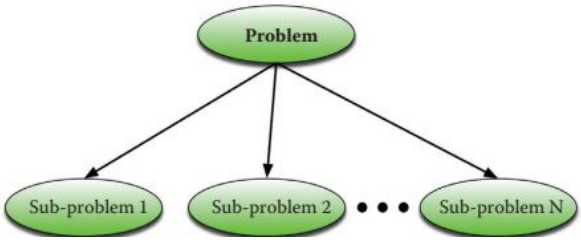
- Conceptualizing, not programming
- Fundamental, not a rote skill
- A way that humans, not computers, think
- Ideas, not artifacts
- For everyone, everywhere

3. Pillars of Computational Thinking:

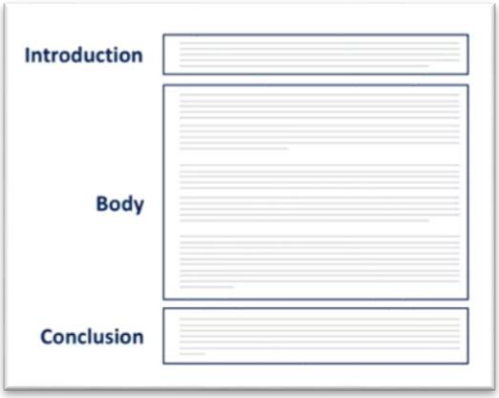


3.1. Decomposition:

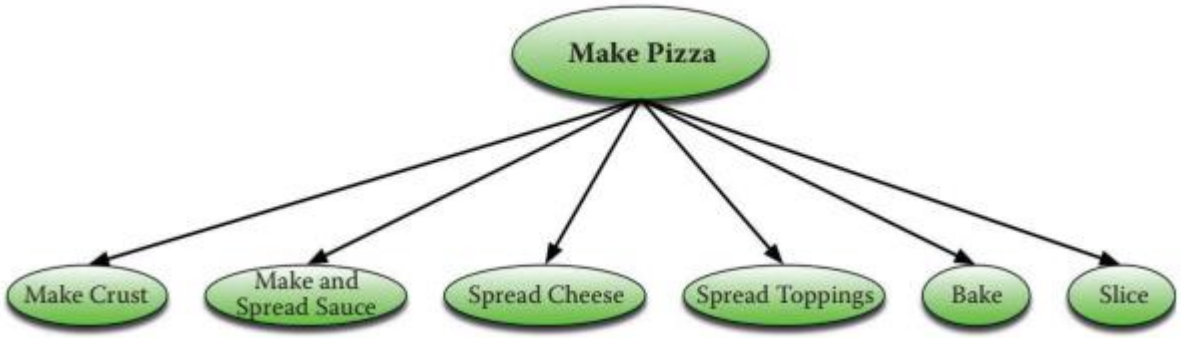
Decomposition is the process of taking a complex problem and breaking it into more manageable sub-problems. Although solving the complex problem as a whole may seem very difficult, the solution to each sub-problem may be much simpler by putting together the solutions to the sub-problems.



Daily Examples of Decomposition: Suppose, you are given the assignment of writing a paper.



Another example



3.2. Pattern Recognition:

When we decompose the problem, we frequently find patterns among the sub-problems, i.e., similarities or shared characteristics. Discovering these patterns make the complex problem easier to solve since we can use the same solution for each occurrence of the pattern.

Example: Which of the following is NOT like the others?

[A] People standing in line at the store (queue)

[B] List of print jobs waiting to be printed (queue)

[C] Set of tennis balls in their container (stack)

[D] Vehicles lined up behind a toll booth (queue)

[E] Patients waiting to see the doctor (queue)

3.3. Data Representation and Abstraction:

Data representation and abstraction involves determining what characteristics of the problem are important and filtering out those that are not. From this, we can create a representation of what we're trying to solve. In other words, A more appropriate definition is that an abstraction is anything that allows us to concentrate on important characteristics while deemphasizing less important, perhaps distracting, details. For example, you are using abstraction if you tell someone that you just saw a red Corvette. In this case the color and model of the car were considered to be the most important, while details of the model year, engine displacement, wheel dimensions, and so forth are omitted.

3.4. Algorithm:

An algorithm is a set of step-by-step instructions of how to solve a problem. It identifies what is to be done (the instructions), and the order in which they should be done.

Example 1: A junior executive for getting out of bed and going to work. Consider the "rise-and-shine algorithm"

(1) Get out of bed,

(2) take off pajamas,

(3) take a shower,

(4) get dressed,

(5) eat breakfast,

(6) carpool to work.

This routine gets the executive to work well prepared to make critical decisions.
Suppose that the same steps are performed in a slightly different order:

(1) Get out of bed,

(2) take off pajamas,

(3) get dressed,

(4) take a shower,

(5) eat breakfast,

(6) carpool to work.

In this case, our junior executive shows up for work soaking wet.

Example 2: Making a cup of tea,

- Put the tea bag in a cup
- Fill kettle with water
- Bring it to a boil
- Pour hot water in a cup
- Steep for 4 minutes
- Remove teabag

Lab Task (Please implement yourself and show the output to the instructor)

1. Think of a problem that could be solved using computational thinking, describe it, and then apply all four pillars of computational thinking to solving that problem.

Lab exercise (submit as a report)

1. Think of a problem that could be solved using computational thinking, describe it, and then apply all four pillars of computational thinking to solving that problem

Policy

Copying from the internet, classmates, seniors, or from any other source is strongly prohibited. 100% marks will be deducted if any such copying is detected.