Grade 6

AI for YOUNG MINDS

FfathomKey

AI Series

Future - Ready kids

Think Smart Build Better

Discover, Learn and Build with AI

NEP 2020 • NCF-SE 2023 • CT & AI Competency Aligned

Young Problem Solver

I solve problems step by step.

Technology Explorer

I explore technology and use it wisely

Responsible Digital Citizen

I use technology safely and kindly

Authored by alumni of IIM Bangalore and University of Texas

FathomKey

AI for Young Mind – Grade 6
First Edition: 2026

Published by: Fathomkey
Website: www.fathomkey.in
Email: info@fathomkey.in
Written by: Nadia Afreen
Editorial Support: Fathomkey Academic Team
Designed and Illustrated by: Nadeemur R. Choudhary
ISBN: 978-81-180798-3-5
Printed in India by:

Curriculum Alignment
This book is part of a grade-wise AI learning programme developed for Indian schools. Its content is informed by the principles of **NEP 2020, NCF** and relevant **CTAI** guidance. It supports the development of computational thinking, problem-solving, digital literacy, creativity and responsible use of Artificial Intelligence across school curricula.

LMS Support
This book is supported by the FATHOMKEY Learning Management System (LMS), which provides digital learning resources, interactive activities, assessments and additional support for students and teachers. LMS access is available through registered schools. For access and assistance, visit https://lms.fathomkey.in or contact info@fathomkey.in.

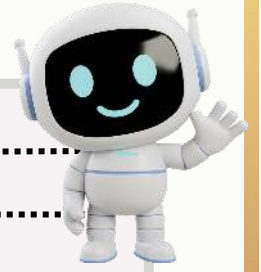
Disclaimer
This book is intended for educational purposes. AI technologies, digital tools, interfaces and features may change over time. Teachers and students should verify important AI-generated information using reliable sources and follow appropriate privacy and online-safety practices. All examples, names, datasets and situations used in this book are fictional or included solely for learning. Product names, logos and trademarks mentioned in the book belong to their respective owners. Their inclusion does not imply endorsement, sponsorship or affiliation.

FATHOMKEY
AI for everyone.

AI for YOUNG MINDS

Grade 6

Discover, Learn and Build with AI



Name: .....

Class: ..... Sec: .....

Subject: ..... Roll No: .....

School: .....

CTAI STUDENT BOOK

FathomKey

Authored by alumni of IIM Bangalore and University of Texas

PREFACE

Artificial Intelligence is changing how people study information, recognise patterns, solve problems and make decisions. To participate confidently in this changing world, students need more than the ability to use digital tools. They must understand how such tools work, examine their outputs carefully and use them responsibly.

AI for Young Minds – Grade 6 develops this understanding through a structured and age-appropriate learning journey. The book begins with computational thinking, logical reasoning, algorithms and flowcharts. Students learn how complex problems can be divided into manageable parts and how clear instructions, conditions and testing contribute to dependable solutions.

The book then introduces Artificial Intelligence by distinguishing it from ordinary software and rule-based automation. Learners explore how AI models learn from examples, recognise patterns in data and produce predictions or suggestions. Practical examples help them understand important ideas such as training data, features, labels, classification, confidence scores, thresholds and human review.

Responsible use is integrated throughout the book. Students learn to protect personal information, create strong passphrases, recognise suspicious messages, verify AI-generated content and consider fairness, bias, accuracy and transparency. They are repeatedly reminded that AI outputs are suggestions—not guaranteed facts—and that people remain responsible for important decisions.

Activities, case studies, exercises and AI Labs encourage students to investigate problems before using technology. Where generative AI tools are introduced, learners first complete their own thinking, use only fictional or teacher-approved information, evaluate the AI response and retain control of the final decision.

The concluding AI Creator Journey guides students in identifying a meaningful problem, deciding whether AI is genuinely required, planning a safe solution, creating a prototype and testing it under different conditions. This process develops logical thinking, creativity, collaboration, communication and responsible digital habits.

We hope this book encourages every learner to question thoughtfully, explore confidently and use Artificial Intelligence with curiosity, care and responsibility.

— **Team Fathomkey**

Inside the Series

🎯 **What You Will Learn**

The section tells you what you will learn in the chapter and the skills you will develop.

🎨 **Activity Corner**

Enjoy hands-on activities, drawing, puzzles, and mini projects to reinforce your learning.

🤖 **Byte's Smart Tip**

Byte shares smart tips to help you to solve problems using Computational Thinking.

🌍 **Real-Life Connection**

Discover how Computational Thinking helps us to solve problems in our daily lives.

📖 **Let's Begin with Introduction**

Every chapter begins with an interesting introduction that connects the lesson to everyday life.

Learn important words and their meanings in simple and easy language.

📘 **Definition**

💻 **AI Connection**

Explore how Artificial Intelligence and smart machines use instructions to complete tasks.

⭐ **Did You Know?**

Read amazing facts that make learning more interesting and fun.

🧠 **Think About It**

Answer thought-provoking questions that encourage observation, reasoning, and discussion.

🏆 **Challenge Yourself**

Apply your learning through HOTS, analytical thinking, and real-life problem-solving activities.

📝 **Exercise Time**

Practice what you have learned through fun and engaging questions.

📚 **Let's Revise**

Quickly revise the chapter using a summary, glossary, and a simple mind map.

CONTENTS

UNIT 1

Chapter 1

Computational Thinking and Logic

Solving Complex Problems with Computational Thinking

"A difficult problem becomes easier when we organise the way we think. "

🎯 Learning Outcome

After completing this chapter, you will be able to:

- ✅ explain how Computational Thinking helps to solve complex problems
- ✅ break a complex problem into smaller, manageable parts
- ✅ identify useful patterns and relationships
- ✅ separate important information from unnecessary details
- ✅ design an organised solution using clear steps
- ✅ recognise constraints and dependencies in a problem
- ✅ compare, test and improve possible solutions
- ✅ explain how Computational Thinking supports the development of AI systems

Chapter Introduction

Every day, we solve problems. Some problems, such as choosing a book to read, may be solved quickly. Other problemsinvolve several people, tasks, rules and decisions.

Imagine that your school is organising an inter-house sports tournament. The organisers must select events, prepare the ground, create team schedules, arrange equipment, assign volunteers, follow safety rules and ensure that two teams are not expected to play in different places at the same time.

Trying to manage everything at once would be confusing. A better approach is to divide the large problem into smaller parts, look for repeated arrangements, focus on important information and prepare a clear plan.

This organised problem-solving approach is called **Computational Thinking.**

Computational Thinking is useful not only in computer science. Students, teachers, doctors, engineers, designers, scientists and AI project teams use similar thinking skills to understand and solve complex problems.

UNDERSTANDING COMPUTATIONAL THINKING

What Is Computational Thinking?

Computational Thinking is an organised way of solving problems by breaking them down, finding patterns, focusing on useful information and designing clear steps.

The word computational does not mean that only computers can use this method. It means that the solution is organised clearly enough to be understood, tested and, in some situations, carried out with the help of a computer.

Computational Thinking helps us answer questions such as:

- What exactly is the problem?
- Which smaller tasks make up the problem?
- Which details are important?
- Have we solved a similar problem before?
- What steps should be followed?
- How will we know whether the solution works?
- What should we improve if it does not work?

> ⭐ **Did You Know?**
> Computer scientists often test a solution several times before accepting it. Finding an error during testing is useful because it shows where the solution needs improvement.

Simple and Complex Problems

A simple problem usually has a small number of steps and conditions.

Example: Arrange five books in alphabetical order.

A complex problem contains several connected parts, choices, people, conditions or possible solutions.

Example: Reorganise the school library so that students can locate books quickly without disturbing ongoing library periods.

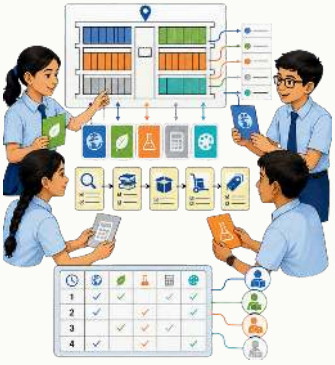
The second problem requires several decisions:

- How should books be grouped?
- Which shelves are available?
- Who will move the books?
- How will book locations be recorded?
- When can the work be completed?
- How will students find the new locations?

A complex problem is not always impossible or extremely difficult. It simply needs careful organisation because many parts are connected.

Computational Thinking Is Not Guessing

Guessing may occasionally produce the correct result, but it does not provide a dependable method. Computational Thinking uses evidence, logic and clear steps.



A well-designed solution should be:

- understandable
- practical
- testable
- repeatable
- open to improvement.

THE FOUR CORE SKILLS

Computer scientists often test a solution several times before accepting it. Finding an error during testing is useful because it shows where the solution needs improvement.

1. Decomposition – Breaking Down the Problem

Decomposition means dividing a large or complex problem into smaller, manageable parts.

Suppose a class is organising an educational exhibition. The project can be divided into:

- *selecting a theme*
- *forming student teams*
- *preparing exhibits*
- *arranging tables and display boards*
- *creating a visitor schedule*
- *checking electrical and safety requirements and*
- *cleaning the venue after the event.*

Each smaller task can then be planned separately.

2. Pattern Recognition – Find Similarities

Pattern recognition means identifying repeated features, similarities, differences or relationships.

The exhibition team may notice that:

- *every exhibit requires a title card*
- *each team needs the same basic materials*
- *visitor numbers are usually highest after lunch and*
- *electrical exhibits need a nearby power point.*

Recognising these patterns can save time and reduce repeated work.

3. Abstraction – Focus on What Matters

Abstraction means selecting the information that is important for solving a problem and leaving out details that are not currently needed.

While preparing the visitor schedule, the organisers need to know:

- *the number of visitors*
- *available time slots*
- *room capacity and*
- *the time required at each exhibit.*

The colour of a visitor's shoes is not relevant to the schedule, so it can be ignored.

Abstraction does not mean removing information carelessly. It means choosing information according to the purpose of the task.

4. Algorithm Design – Creating Clear Steps

OAn algorithm is an ordered set of clear steps used to complete a task or solve a problem.

An algorithm for admitting visitors might be:

1. *Check the visitor's time slot.*
2. *Confirm that space is available.*
3. *Give the visitor a route card.*
4. *Direct the visitor to the first exhibit.*
5. *Record the visitor's entry.*
6. *Repeat the process for the next visitor.*

CONSTRAINTS, DEPENDENCIES AND SUCCESS CRITERIA

A solution cannot be planned by looking only at the desired result. A problem solver must also identify what may limit the plan, which tasks depend on other tasks and how success will be measured.

Constraints

A **constraint is a limit or condition that a solution must follow.**

Common constraints include:

- limited time
- limited money
- available materials
- number of people
- space
- safety requirements
- school rules.

Thinking Skills HOTS!!

Two teams are asked to prepare 20 display charts.

- Team A starts decorating immediately.
- Team B first confirms the topics, available materials, deadline and size of each display area.

Which team is using Computational Thinking more effectively?

Example: A class has ₹1,500 and two days to prepare a science display. The budget and time are constraints.

Constraints do not always prevent a solution. They help define what kind of solution is practical.

Dependencies

A dependency exists when one task must be completed before another task can begin.

For **example:**

- The exhibition theme must be selected before materials are purchased.
- The number of teams must be known before tables are assigned.
- A model must be tested before it is displayed.
- Permission must be received before photographs are taken.

Ignoring dependencies may cause delays, repeated work or unsafe decisions.

Success Criteria

Success criteria are the conditions used to judge whether a solution has achieved its purpose.

For a new library arrangement, success criteria might include:

- students can locate a book in less than three minutes
- every shelf has a clear category label
- the book-location record is accurate and
- pathways remain clear and safe.

Without success criteria, a team may complete many tasks but still not know whether the main problem was solved.

Remember

A creative idea becomes a practical solution only when it respects real limits, follows the correct order and achieves measurable results.

AI Connection

AI project teams also work with constraints and dependencies. For example, an image-based plant-health system needs suitable photographs before a model can be trained. The model must be tested before its suggestions are shown to users.

Byte's Smart Tip

A strong solution usually uses all 4 skills together. Breaking down a problem is helpful, but the smaller parts must still be connected through an organised plan.

A COMPUTATIONAL THINKING PROCESS

The four core skills can be used through a six-stage problem-solving process.

Stage 1: Understand the Problem

State what is happening, who is affected and what result is required.

Stage 2: Decompose the Problem

Divide the main problem into smaller tasks or questions.

Stage 3: Find Patterns and Relationships

Look for repeated events, similarities, differences and connections.

Stage 4: Focus on Relevant Information

Select the details needed for the solution. Identify constraints and dependencies.

Stage 5: Design the Solution

Create clear steps. Assign responsibilities and decide how the parts will work together.

Stage 6: Test, Evaluate and Improve

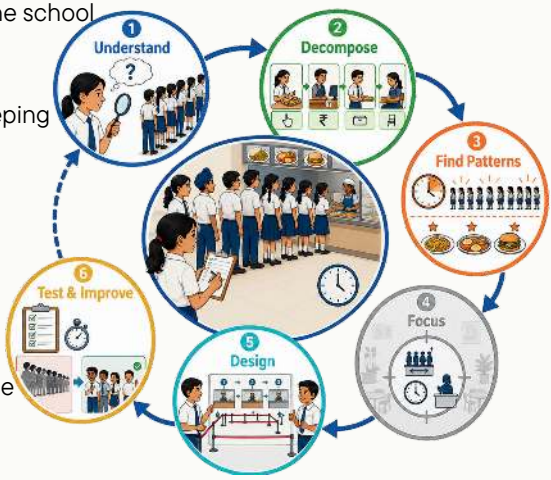
Try the solution, compare the result with the success criteria and revise the plan if necessary.

This process is iterative. An iterative process allows the problem solver to return to an earlier stage and make improvements.

PWorked Example — Reducing the Lunch Queue

The situation

Students spend too much time waiting at the school canteen during lunch break.



Understand

The goal is to reduce waiting time while keeping food distribution safe and fair.

Decompose

The team investigates:

- when students arrive
- how orders are placed
- how payment is handled
- where food is collected
- how many serving counters are available
- which items take the longest to serve.

Find patterns

The team observes that:

- most students arrive during the first ten minutes
- students often decide what to buy only after reaching the counter
- a few popular items cause the longest delays and
- separate payment and collection activities slow the line.

Focus

Important information includes arrival time, order type, service time and number of counters. The colour of a student's schoolbag is irrelevant.

Design

The team proposes:

1. Display the menu before lunch.
2. Allow students to choose an item in advance.
3. Create separate collection points for pre-packed and freshly prepared items.
4. Place clear queue markers.
5. Assign a teacher to supervise the trial.

Test and improve

The plan is tested for three days. Waiting time becomes shorter, but one collection point becomes crowded. The team changes the position of the counters and tests the plan again.
The first solution did not need to be perfect. Testing provided evidence for improvement.

Chapter Check-Up

A. Fill in the blanks.

1. Dividing a large task into smaller parts is called _____.
2. A repeated feature or relationship is a _____.
3. A time limit placed on a project is a _____.
4. Selecting only relevant information is called _____.

B. True or False

1. Computational Thinking can be used only while programming.
2. A complex problem may contain several connected tasks.
3. Abstraction means ignoring all details.
4. Testing helps a team identify possible improvements.

C. Name the Skill

Identify the main Computational Thinking skill used in each situation.

- Separating a school trip plan into transport, food, permission and safety.
- Noticing that the school gate becomes crowded at the same time each day.
- Using only distance, travel time and road conditions while selecting a route.
- Writing ordered steps for borrowing and returning sports equipment.
- Revising an arrangement after a trial reveals a problem.

D. Apply Your Thinking

Your class wants to create a system for managing lost-and-found objects.
Discuss:

1. What smaller tasks make up the problem?
2. Which information should be recorded for each object?
3. Which information would be unnecessary?
4. What rules should be followed before returning an object?
5. How would you test whether the system works?

E. Quick Reflection

Complete the statement:
“The part of Computational Thinking that I use most often is _____ because _____.”

COMPUTATIONAL THINKING IN ACTION

Case Study 1 — Organising an Inter-house Sports Tournament

A school wants to organise a two-day tournament involving four houses, six sports and three playing areas.

Decomposition	Pattern Recognition	Abstraction	Algorithm Design
The organising team divides the work into: • event registration • team verification • match scheduling • equipment management • ground preparation • score recording • first aid • refreshments and • prize distribution.	The team notices that every match needs: • two teams • a playing area • a referee • equipment • a fixed duration and • a score record. A common match template can therefore be reused.	For scheduling, the team focuses on: • team names • event • duration • venue and • availability. Uniform colour and team slogans are not required for preparing the schedule.	Algorithm Design 1. List all registered teams. 2. Group teams by sport. 3. Check the available grounds and time slots. 4. Assign one match to each available slot. 5. Check that no team has two matches at the same time. 6. Add rest periods. 7. Ask the sports teacher to review the schedule. 8. Correct conflicts and publish the final version.

Testing and improvement

A review shows that one team has matches in consecutive time slots at distant grounds. The schedule is revised to include adequate travel and rest time.

Case Study 2 – Planning Watering for the School Garden

he school garden contains flowering plants, herbs, vegetables and young trees. Watering every plant in exactly the same way wastes water and may harm some plants.

Breaking down the problem

The gardening team considers:

- plant type
- soil condition
- sunlight
- recent rain

- plant age
- watering frequency and
- available water.

Finding patterns

Plants in sunny areas dry faster. Young plants need more frequent checking. Soil remains wet longer after rainfall.

Focusing on relevant information

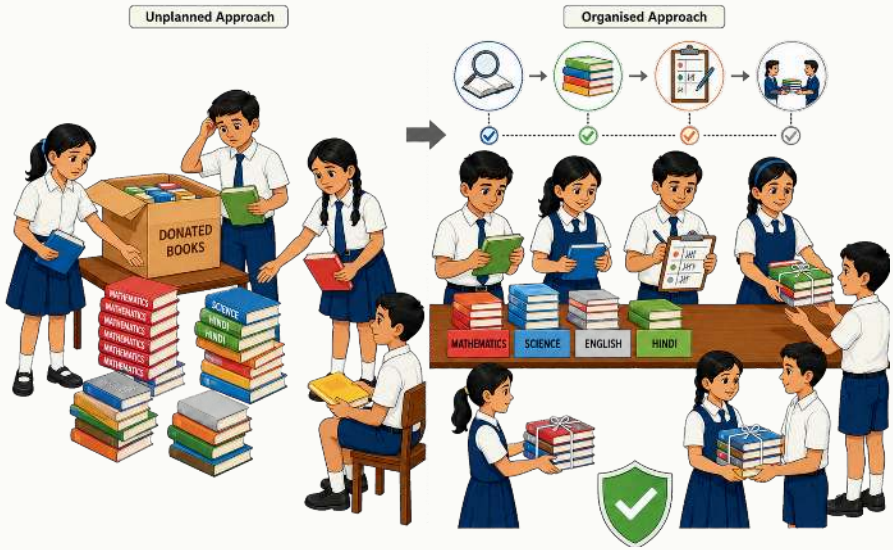
The team needs plant type, soil moisture, weather and location. The colour of the gardening tools does not affect watering decisions.

Designing a simple decision process

- Check whether it rained recently.
- Observe the soil near the plant.
- Identify the plant group.
- If the soil is dry, use the recommended amount of water.
- If the soil is still moist, check it again later.
- Record unusual plant conditions for the gardener.

Human responsibility

The process supports students, but the school gardener makes the final decision when a plant appears unhealthy or the weather is unusual.



COMPARING AND IMPROVING SOLUTIONS

A complex problem may have more than one possible solution. Computational Thinking helps us compare alternatives instead of accepting the first idea.

The Problem

A school has received 120 donated books. The books must be checked, recorded and distributed among four classroom reading corners.

Approach A

Start Immediately

Students carry books to different classrooms as soon as the boxes arrive. They decide where each book belongs while unpacking.

Possible problems:

- some classrooms may receive too many books
- similar books may be separated
- damaged or duplicate books may not be recorded
- book counts may be inaccurate and
- work may have to be repeated.

Approach B

Use an Organised Process

1. Count the books.
2. Check their condition.
3. Group them by reading level and subject.
4. Record each group.
5. identify the needs of each classroom.
6. decide how many books each classroom should receive.
7. distribute the books.
8. verify the final count.

Criteria for Comparing Solutions

Criterion	Question to Ask
Accuracy	Does the solution produce the correct result?
Efficiency	Does it avoid unnecessary time, effort and materials?
Fairness	Are resources distributed using reasonable criteria?
Safety	Does the solution protect people and property?
Clarity	Can another person understand and follow the process?
Adaptability	Can the plan be adjusted if conditions change?

Approach B is likely to produce a more dependable result because it organises the information before distribution.

However, Approach B could still be improved. For example, students could create one common recording sheet so that all groups use the same categories.

What Makes a Good Solution?

A good solution is not simply the fastest solution. It should meet the purpose of the task while respecting its constraints and success criteria.

Think and Explain

A fast solution produces several errors. A slower solution produces an accurate result but misses the deadline.

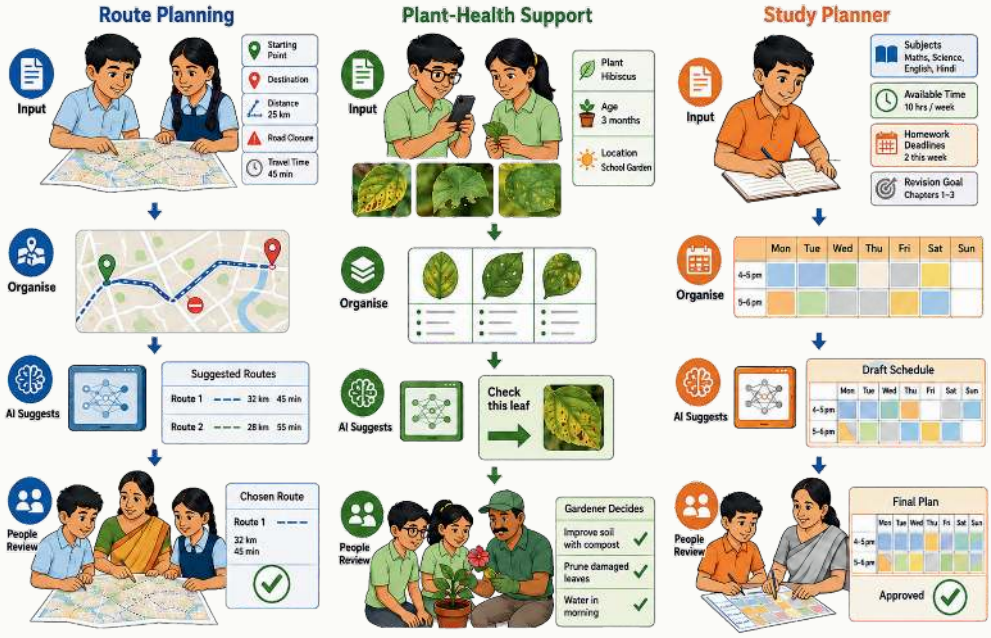
Is either solution completely successful?

What changes could balance speed and accuracy?

HOW COMPUTATIONAL THINKING SUPPORTS AI

Computational Thinking and Artificial Intelligence are connected, but they are not the same.
Computational Thinking is a problem-solving approach.
Artificial Intelligence is a technology that enables machines to perform certain tasks that usually require human intelligence.

A project does not become an AI project merely because its steps are organised. However, AI teams use Computational Thinking to design the complete system around an AI model.



Example 1 — Navigation Application

A navigation application must handle several smaller tasks:

1. identify the starting point and destination
2. compare possible routes
3. recognise traffic patterns
4. consider road closures
5. estimate travel time
6. update the route when conditions change.

The application focuses on relevant information such as location, distance, traffic and road conditions. It does not need unrelated details such as the colour of nearby buildings.

Example 2 — Plant-health Support System

An image-based system that suggests whether a plant may need attention involves:

1. receiving a suitable plant photograph
2. checking image quality
3. focusing on useful visual features
4. comparing the features with learned patterns
5. producing a possible category
6. showing the suggestion to a gardener and
7. recording whether the suggestion was useful.

The AI model provides a suggestion. A knowledgeable person reviews the plant and decides what action to take.

Example 3 — AI-assisted Study Planner

A student may ask a generative AI tool to suggest a study plan. A useful request must include relevant constraints:

- subjects to study
- available time
- test dates
- rest breaks and
- difficult topics.

The student must then check whether the suggested plan is realistic and revise it when necessary.

Computational Thinking Across an AI Project

Computational Thinking Skill	Use in an AI Project
Decomposition	Divides the system into data collection, model task, interface, review and monitoring
Pattern Recognition	Helps identify useful relationships in examples
Abstraction	Selects relevant features and removes unnecessary information
Algorithm Design	Organises the steps followed by software, AI and people
Testing and Improvement	Checks outputs and improves the system when problems appear

AI Explorer

Choose one AI-supported application that you have seen or used. Investigate:

- What problem does it try to solve?
- What smaller tasks may be involved?
- What information is important?
- What output does it provide?
- Who should check or act on the output?

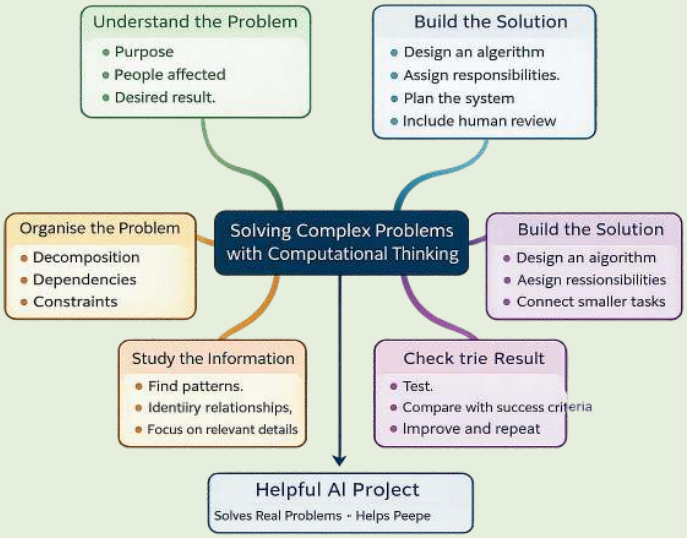
📝 Revision Summary

- Computational Thinking is an organised approach to solving problems.
- Complex problems can be divided into smaller and more manageable parts.
- Pattern recognition helps identify repeated features and relationships.
- Abstraction helps us focus on information that is relevant to the purpose.
- An algorithm gives clear and ordered steps for completing a task.
- Constraints, dependencies and success criteria affect how a solution is designed.
- Solutions should be tested, evaluated and improved.
- Computational Thinking helps people design effective AI-supported systems.

📖 Glossary

Abstraction: Selecting the information that is important for solving a problem and leaving out details that are not currently needed.
Algorithm: An ordered set of clear steps used to complete a task or solve a problem.
Complex problem: A problem containing several connected parts, choices, people, conditions or possible solutions.
Computational Thinking: An organised way of solving problems by breaking them down, finding patterns, focusing on useful information and designing clear steps.
Constraint: A limit or condition that a solution must follow.
Decomposition: Dividing a large or complex problem into smaller, manageable parts.
Dependency: A relationship in which one task must be completed before another task can begin.
Iteration: Repeating a process to test and improve a solution.
Pattern recognition: Identifying repeated features, similarities, differences or relationships.
Success criteria: Conditions used to judge whether a solution has achieved its purpose.

Mind Map



Exercise

A. Fill in the Blanks

1. A problem containing several connected parts is called a _____ problem.
2. _____ means identifying similarities and relationships.
3. A limit that a solution must follow is called a _____.
4. Conditions used to evaluate a solution are called _____.
5. A solution that is repeatedly tested and improved is _____.
6. A clear, ordered set of instructions is called an _____.

B. State Whether True or False

1. Computational Thinking can be used only while writing computer programs.
2. Abstraction helps us focus on relevant information.
3. Every dependency is a problem that must be removed.
4. The first solution to a problem is always the most efficient.
5. AI projects can benefit from Computational Thinking.
6. Testing may reveal that an earlier step needs to be changed.

C. Match the Following

Column A	Column B
1. Decomposition	a. Measuring whether the goal was achieved
2. Pattern Recognition	b. A limit affecting the solution
3. Abstraction	c. Dividing a problem into parts
4. Constraint	d. Finding repeated relationships
5. Success Criteria	e. Selecting important information

D. Multiple-Choice Questions

1. Which action is an example of decomposition?
   a. Selecting the cheapest solution
   b. Dividing an event into registration, scheduling and equipment tasks
   c. Ignoring all available information d. Repeating the same plan without testing
2. Which information is most relevant when planning a bus timetable?
   a. Colour of the driver's shoes b. Favourite subject of each passenger
   c. Route duration and expected passenger numbers
   d. Design of the classroom noticeboard
3. Which statement best describes a dependency?
   a. A task that is not important b. A task that must wait for another task
   c. A repeated visual design d. A solution that has no constraints
4. Why are success criteria important?
   a. They make every solution identical. b. They guarantee that no mistake will occur.
   c. They remove the need for testing.
   d. They help determine whether a solution meets its goal.
5. A student tests an algorithm, finds an error and changes two steps. This is an example of:
   a. abstraction. b. iteration. c. decoration. d. prediction

A. Short-Answer Questions

1. What is Computational Thinking?
2. What makes a problem complex?
3. Explain the difference between a constraint and a dependency.
4. Why is abstraction useful?What is an iterative process?
5. Why should solutions be compared using the same criteria?
6. How does Computational Thinking support the development of AI systems?

B. Long-Answer Questions

1. Explain the four main Computational Thinking skills using one connected example.
2. Describe the six-step process for solving a complex problem.
3. Explain why the fastest solution may not always be the best solution.

C. Think and Explain

1. A school introduces an online form for borrowing sports equipment. The form works correctly, but many students cannot access it during the sports period. Is the solution successful? Explain using suitable success criteria.
2. A student removes all unusual observations from a science investigation because they do not match the expected pattern. Is this a correct use of abstraction? Why or why not?

D. Real-Life Problem-Solving Task

Your school library becomes crowded during the lunch break. Students need to:
return books,borrow books,ask the librarian for help,use reading tables, search for books.
Apply Computational Thinking to the problem Include:

1. a clear problem statement
2. at least four smaller parts
3. two useful patterns that could be investigated
4. relevant and irrelevant information
5. two constraints and two dependencies
6. a possible algorithm
7. three success criteria
8. a method for testing the solution.

E. Byte's Challenge

A school wants to reduce electricity waste in classrooms.
Two suggestions are offered:

- Switch off every electrical device at exactly 2:00 p. m.
- Check whether a room is occupied and switch off unnecessary devices when the room is empty.

Compare the two approaches. Identify their assumptions, advantages, risks and limitations. Then propose an improved solution that keeps people responsible for the final action.

AI Lab: Human-First Planning with a Generative AI Tool

Activity Goal

Use Computational Thinking to prepare a plan, compare it with an AI-generated suggestion and create an improved final solution.

Activity 1 – Design a plan for a Low-Waste Class Event.

The event may include displays, games, food, invitations and student presentations. The plan should reduce unnecessary paper, food waste, disposable materials and electricity use.

Tools Required: **ChatGPT,Google Gemini,Claude(teacher-approved generative AI tool)**

Part 1: Think Before Using AI

Work individually or in a group.

Write a clear problem statement.

Divide the event into smaller parts.

- Identify possible patterns from earlier class events.
- List relevant information.
- Identify at least three constraints.
- Identify two dependencies.
- Write success criteria.

Create your own first plan. (***Do not use the AI tool during this part.)***

Part 2: Prepare a Safe and Clear Prompt

Use a **prompt** such as:

We are Grade 6 students planning a **two-hour low-waste class** event for 35 students. The event includes project displays, two games and simple refreshments. We have a limited budget and must follow school safety rules. Suggest an organised plan that reduces paper, food waste, disposable materials and unnecessary electricity use. Divide the plan into preparation, event-time and clean-up tasks. Do not request personal information.

Do not include:

- student names
- phone numbers
- home addresses
- photographs
- account details
- private school records.

Part 3: Examine the AI Response

Check the response using these questions:

1. Did it understand the problem?
2. Did it divide the work into useful parts?
3. Did it consider the stated constraints?
4. Are the steps in a sensible order?
5. Are any suggestions unsafe or unrealistic?
6. Did it make assumptions that were not provided?
7. Which suggestions are useful?
8. Which suggestions should be changed or removed?

Part 4: Compare the Plans

Prepare a comparison table.

Feature	Our First Plan	AI Suggestion	Final Decision
Decomposition			
Constraints			
Dependencies			
Clear steps			
Waste reduction			
Safety			
Practicality			

Part 5: Create the Improved Plan

Combine useful ideas, correct weak suggestions and prepare a final plan.

Your final plan must include:

- event tasks
- assigned roles
- required materials
- ordered steps
- waste-reduction actions
- safety checks
- success criteria
- a clean-up process.

Reflection Questions

1. What did your group identify that the AI tool missed?
2. Which AI suggestion improved your original plan?
3. Did the AI make any incorrect assumptions?
4. How did Computational Thinking help you evaluate the response?
5. Why should people make the final decision?
6. How would you improve your prompt next time?

Responsible AI Reminder

AI can suggest possibilities, but students and teachers must check whether those possibilities are correct, safe, fair and practical.