### **Day 1: Introduction to Logical Thinking(intro and puzzle all day)**

* **Objective:** Introduce students to the concept of logical thinking and problem-solving.
* **Activities:**
  + **Discussion:** Explain what logical thinking is and why it’s important in computer science.
  + **Puzzle Activity:** Classic puzzles like the “River Crossing Puzzle” or “The Fox, Chicken, and Grain”, “3 litre and 4 litre cup problem” problem.
  + **Debrief:** Discuss strategies used to solve the puzzles and how they apply to programming.

### **Day 2: Basic Algorithms and Flowcharts**

* **Objective:** Teach students the basics of algorithms and how to visualize them using flowcharts.
* **Activities:**
  + **Lecture:** Introduction to algorithms and flowcharts.
  + **Activity:** Create flowcharts for simple problems (e.g., making a cup of tea).
  + **Hands-on:** Convert the flowchart into a pseudocode.

### **Day 3: Introduction to Programming Concepts**

* **Objective:** Introduce basic programming concepts through simple exercises.
* **Activities:**
  + **Lecture:** Variables, data types, and basic operations.
  + **Activity:** Write simple programs in a beginner-friendly language (e.g., Python) to perform basic arithmetic operations.
  + **Exercise:** Students write a program to calculate the area of a rectangle.

### **Day 4: Conditional Statements and Logic**

* **Objective:** Teach conditional statements and logical operators.
* **Activities:**
  + **Lecture:** Explain if, elif, else, and logical operators (AND, OR, NOT).
  + **Activity:** Solve logic puzzles that require conditional statements (e.g., “If you’re not in the first place, you’re in the last place.”).
  + **Programming Exercise:** Write a program to determine if a number is odd or even.

### **Day 5: Loops and Iteration**

* **Objective:** Introduce loops and iterative processes.
* **Activities:**
  + **Lecture:** Explain for and while loops.
  + **Activity:** Puzzles that involve repeating steps (e.g., “How many steps to reach the 100th floor?”).
  + **Programming Exercise:** Write a program to print a multiplication table.

### **Day 6: Functions and Modularity**

* **Objective:** Teach the concept of functions and modular programming.
* **Activities:**
  + **Lecture:** Explain functions, parameters, and return values.
  + **Activity:** Solve problems that can be broken into smaller functions (e.g., “Write a function to calculate the factorial of a number.”).
  + **Programming Exercise:** Refactor a previous program by breaking it into functions.

### **Day 7: Debugging and Problem Solving**

* **Objective:** Develop debugging skills and problem-solving strategies.
* **Activities:**
  + **Lecture:** Introduction to common bugs and debugging techniques.
  + **Activity:** Provide students with buggy code and have them debug it.
  + **Programming Exercise:** Students fix errors in a provided piece of code and test it.

### **Day 8: Integrating Concepts and Final Challenges**

* **Objective:** Integrate all concepts learned and apply them to more complex problems.
* Teaching them how to actually learn programming and what is the correct way
* **Activities:**
  + **Group Activity:** Solve a larger, multi-step problem that requires all concepts learned (e.g., a simple text-based game or a number-guessing game).
  + **Discussion:** Reflect on the problem-solving process and how logical thinking played a role.
  + **Review:** Quick recap of key concepts and Q&A session.