

Solution Sheet – Analysing and Solving Problems Lab Exercise

The explanations provide a simple and structured way to support understanding of abstract thinking and computational thinking.

Exercise 1

Figure out whether it is abstract or concrete thinking. Provide justifications for your selection.

a. Comparing your implementation with one of your friends and checking which one is user friendly

Concrete Thinking – This focuses on directly observable features such as appearance, usability, and practical results.

b. Doing a Google search to find similar approaches before writing code

Abstract Thinking – This involves analysing ideas, identifying patterns, and learning general approaches before implementation.

c. Explaining a complex theory using metaphors and analogies

Abstract Thinking – Analogies simplify complicated concepts into understandable models.

d. Replicating a similar project completed earlier

Concrete Thinking – This depends on previous direct experience and copying an existing solution.

e. Researching competitors and product features before building a component

Abstract Thinking – This requires analysing trends, identifying common patterns, and understanding high-level requirements.

Exercise 2

You are asked to integrate a new software component into an existing student management system.

Abstract Thinking Approach:

An abstract thinker would first focus on understanding the overall problem rather than the specific code. They would identify the main requirements, analyse why the existing component failed, and look for common patterns between the old and new systems. The thinker would separate the important features from unnecessary details and design a flexible solution that can integrate easily with the student management system.

Steps may include:

- Identifying system requirements
- Finding mismatches between the component and requirements
- Designing reusable interfaces or modules

- Testing integration with the existing system

Concrete Thinking Comparison:

A concrete thinker would focus more on the immediate practical task. Instead of redesigning the system, they may directly modify the current code and fix errors one by one. This can solve short-term issues but may not create a scalable or reusable solution.

Exercise 3

Sum up Fibonacci numbers from 1 to 1000. Explain how computational thinking can be applied here.

Fibonacci sequence:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987

Sum = 2583

Computational Thinking Process:

1. Decomposition – Break the problem into smaller steps such as generating Fibonacci numbers and calculating the sum.
2. Pattern Recognition – Observe that each Fibonacci number is the sum of the previous two numbers.
3. Abstraction – Ignore unnecessary details and focus only on numbers less than or equal to 1000.
4. Algorithm Design – Create a step-by-step procedure:
 - Start with 1 and 1
 - Generate the next number using the previous two
 - Continue until the value exceeds 1000
 - Add all generated numbers together

Exercise 4 – Pattern Printing

Pattern:

1
12
123
1234
12345
123456

Computational Thinking Process:

1. Decomposition – Separate the task into rows and columns.
2. Pattern Recognition – Each row starts from 1 and increases up to the row number.
3. Abstraction – Focus only on the rule that controls the numbers.
4. Algorithm Design – Use nested loops:
 - Outer loop controls the row number
 - Inner loop prints numbers from 1 up to the row value

Exercise 5 – Bomb Detection Game Board

Abstract Solution Using Computational Thinking:

1. Decomposition:

- Divide the board into smaller squares.
- Check each square individually.

2. Pattern Recognition:

- Bombs are hidden in certain squares.
- The same checking process is repeated for all squares.

3. Abstraction:

- Represent the board as a grid.
- Use simple symbols such as:
B = Bomb
0 = Empty square

4. Algorithm Design:

- Start from the first square.
- Move through every row and column.
- If a bomb is found, increase the bomb counter.
- Continue until all squares are checked.
- Print the total number of bombs.