Activity 1

1. Solving a sliding 15-puzzle

Recursive Decomposition, Exploratory Decomposition

**Recursive Decomposition**

Explanation: The sliding 15-puzzle can be broken down recursively by considering each move as a step that reduces the problem size. At each step, the puzzle can be divided into smaller subproblems, each representing a state of the puzzle after a move. This process continues until the puzzle reaches the solved state.

**Exploratory Decomposition**

Explanation: This technique involves exploring all possible moves and states of the puzzle to find the solution. It is useful in problems where multiple paths or solutions exist, and the goal is to explore different possibilities.

1. Find the frequency of usage of {'ch','de','des','th','es', 'ci'} in the sliding 15-puzzle Wikipedia page

Input Data Decomposition, Output Data Decomposition

**Input Data Decomposition**

Explanation: The problem can be divided based on the input data, which is the text of the Wikipedia page. The text can be partitioned into smaller segments, and each segment can be processed in parallel to find the frequency of the specified substrings.

**Output Data Decomposition**

Explanation: The problem can also be divided based on the expected output, which is the frequency count of each substring. Each substring's frequency can be computed independently.

1. Binary search

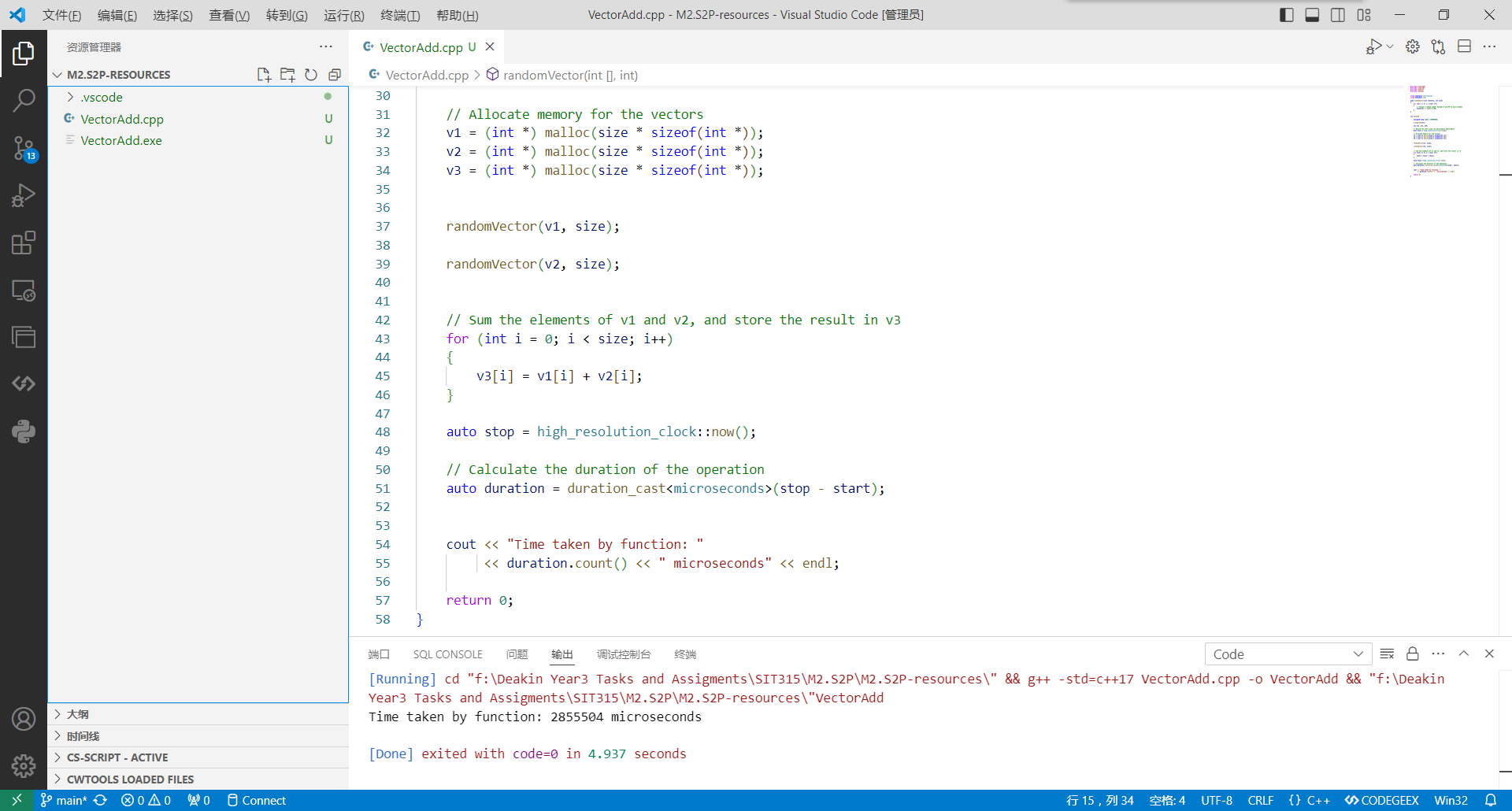
Recursive Decomposition

**Recursive Decomposition**

Explanation: Binary search is inherently a recursive process. At each step, the problem (searching an element in a sorted array) is divided into smaller subproblems by halving the array.

Activity 2

1.2.



3.

**Problem Decomposition:**

1. Data Initialization:

Random vector generation for v1 and v2.

1. Computation:

Summing the elements of v1 and v2 to produce v3.

1. Measurement:

Timing the execution of the tasks.

**Sub-tasks Identification:**

**Parallelizable Tasks:**

Random Vector Generation:

Both vectors v1 and v2 can be generated in parallel.

Vector Summation:

Summing elements of v1 and v2 to produce v3 can be divided into independent chunks and processed in parallel.

**Sequential Tasks:**

Memory Allocation:

Allocate memory for vectors v1, v2, and v3 (needs to be done before using them).

Timing Measurement:

Start and stop timing should encapsulate the parallel operations.

Memory Deallocation:

Freeing allocated memory.

**Implementation Roadmap:**

Step 1: Memory Allocation

Allocate memory for v1, v2, and v3. This step is sequential.

Step 2: Parallel Random Vector Generation

Generate v1 and v2 in parallel.

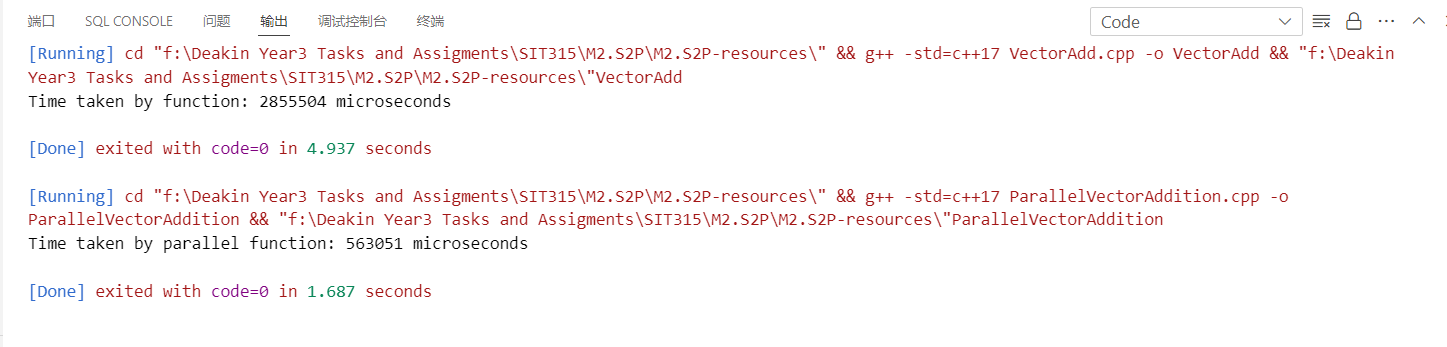
Step 3: Parallel Vector Summation

Divide the summation task into chunks that can be processed in parallel.

Step 4: Memory Deallocation

Free the allocated memory for v1, v2, and v3. This step is sequential.

4.5.



6.