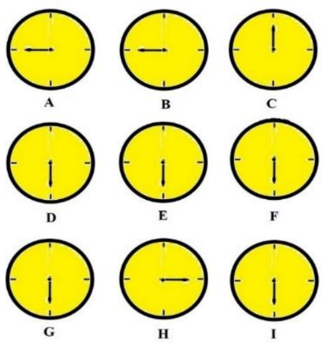
Computing Algorithms

Assignment #5

Solve the following problem

Problem 1: In this problem, there are nine clocks in a 3*3 array shown in figure. The goal is to return all the dials to 12 o'clock with as few moves as possible. There are nine different allowed ways to turn the dials on the clocks. Each such way is called a move. Select for each move a number 1 to 9. That number will turn the dials 90o clockwise on those clocks which are affected according to figure. Write a parallel algorithm using 3 processors to implement this job. Find the speed up and efficiency of your algorithm.



Input Data

Read nine numbers from the INPUT.TXT file.

These numbers give the start positions of the dials. 12=0 o'clock, 1=3 o'clock, 2=6 o'clock, 3=9 o'clock .The example in figure gives the following input data file:

Output Data

Write to the OUTPUT.TXT file with the shortest sequence of moves (numbers), which returns all the dials to 12 o'clock. In case there are many solutions, only one is required. In this example the OUTPUT.TXT file could look as follows 5849:

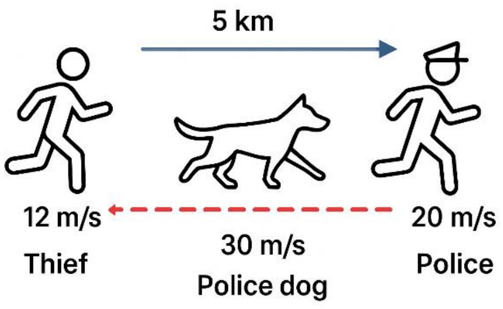
Move	Affected clocks
1	ABDE
2	ABC
3	BCEF
4	ADG
5	BDEFH
6	CFI
7	DEGH
8	GHI
9	EFHI

Example of Method

Each number represents a time according to the following table:

0 =	12 o'clock
1 =	3 o'clock
2 =	6 o'clock
3 =	9 o'clock

Problem 2: A thief escapes from prison, running at a constant speed of 12 m/s.
When the thief has built a 5 km lead, the police begin pursuit from the prison at 20 m/s.
Simultaneously, a police dog starts running at 30 m/s in a continuous pattern: it sprints from the police toward the thief, instantly turns upon reaching the thief and runs back to the police, and repeats this back-and-forth motion until the thief is caught.



Using only a geometric method (e.g., a *space-time diagram*), determine the total distance the dog runs in the forward direction—that is, the sum of only those segments where it moves from the police toward the thief.

Computing Algorithms

Problem 3: You are given a very large file containing millions of integers. The size of the file far exceeds the available RAM, so you cannot load the entire file into memory at once. Your task is to implement an External Merge Sort to sort the file efficiently using limited memory. This is a classic real-world problem used in:
Databases - Operating systems - Big Data platforms (Hadoop, Spark) -Search engines
Write a program that sorts a very large file of integers using external sorting. You must assume the memory available to your program is very limited (e.g., 4 MB only).
Your program must follow these steps:

Requirements

✓ Your program must not use more memory than allowed.
✓ You must delete temporary run files at the end.
✓ Your implementation must handle **very large data sets** (tens of millions of integers).
✓ You may use **Python, C, Java**, or any approved language.

What Students Must Submit

1. **Source code** of the external sort implementation.
2. **README** describing:
   - Memory constraints assumed
   - Chunk size used
   - Merge strategy
   - Time complexity and I/O analysis
3. A **short report** (1–2 pages) explaining:
   - Why external sorting is needed
   - How the algorithm works
   - Possible optimizations (buffering, heap for merge)

Problem 4: Describe the fastest algorithm you can for computing the area of a simple polygon P, given as a list of vertices v1, v2, …, vn in counterclockwise order of occurrence. Prove the correctness of your algorithm and analyze its time complexity.