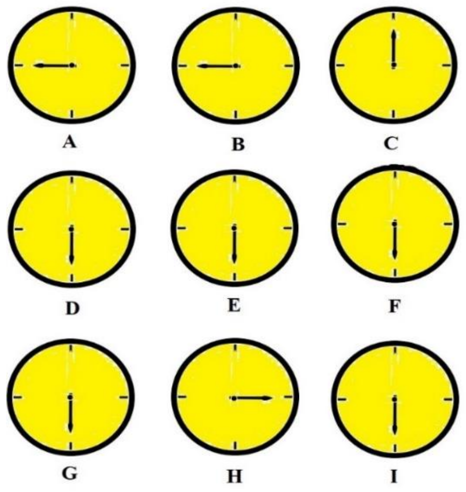
Program: Intelligent Systems – and
Level: **Term: Fall 2022/2023**
Course Code: **(01414-24-022) Concurrent Algorithms & Data Structures**

Assignment #3 Total points: 20

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:

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

Output Data

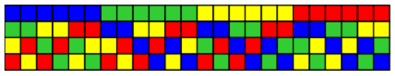
Write to the OUTPUT.TXT file the shortest sequence of moves (numbers), which returns all the dials to 12 o'clock. In case there are many solutions, only one is required. In this example the OUTPUT.TXT file could look as follows 5849:

Example of Method

Each number represents a time according to following table:
0 = 12 o'clock
1 = 3 o'clock
2 = 6 o'clock
3 = 9 o'clock

```
3 3 0         3 0 0         3 0 0          0 0 0         0 0 0
2 2 2   5->   3 3 3   8->   3 3 3   4 ->   0 3 3   9->   0 0 0
2 1 2         2 2 2         3 3 3          0 3 3         0 0 0
```

Problem 2: Given an array of numbers and queries specifying two segments of equal size, how can you quickly check if they are permutations of one another ($n \leq 10^4$)? Write a recursive function that generates all permutations of a given two number n C r. Then write a parallel/concurrent algorithm to do the job. Find the speed up and efficiency of your algorithm. The number of processors can be any number from ≤ n.



https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1206/lectures/procedural-recursion/

Program: Intelligent Systems – and
Level: **Term: Fall 2022/2023**
Course Code: **(01414-24-022) Concurrent Algorithms & Data Structures**

Assignment #3 Total points: 20

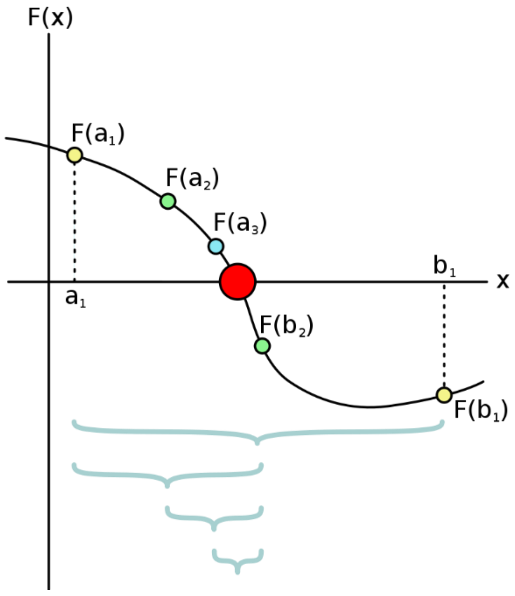
Problem 3: In mathematics, the bisection method is a root-finding method that applies to any continuous function for which one knows two values with opposite signs. The method consists of repeatedly bisecting. The interval Defined by these values and then selecting the subinterval in which the function changes sign, and therefore must contain a root. It is a very simple and robust method, but it is also relatively slow. Because of this, it is often used to obtain a rough approximation to a solution which is then used as a starting point for more rapidly converging methods. The method is also called the interval halving method, the binary search method, or dichotomy method.

Sequential Algorithm: The bisection method may be written in pseudocode as follows:

```
INPUT: Function f,
      endpoint values a, b,
      tolerance TOL,
      maximum iterations NMAX
CONDITIONS: a < b,
          either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0
OUTPUT: value which differs from a root of f(x) = 0 by less than TOL

N ← 1


while N ≤ NMAX do // limit iterations to prevent infinite loop
   c ← (a + b)/2 // new midpoint
   if f(c) = 0 or (b – a)/2 < TOL then // solution found
      Output(c)
      Stop
   end if
   N ← N + 1 // increment step counter
   if sign(f(c)) = sign(f(a)) then a ← c else b ← c // new interval
end while
Output ("Method failed.") // max number of steps exceeded
```



The complexity of the sequential algorithm is log2((b-a)/TOL)
Write parallel/concurrent algorithm to solve this problem? Write a general algorithm by using N processors? Find the speed up and efficiency of your algorithm.

https://www.youtube.com/watch?v=OzFuihxtbtA
chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.iosrjournals.org/iosr-jm/papers/Vol11-issue4/Version-2/F011423236.pdf
chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.cambridge.org/core/services/aop-cambridge-core/content/view/1CE0E551BC6CCC94B3A72FB0B2E59CB7/S1461157015000236a.pdf/div-class-title-a-parallel-root-finding-algorithm-div.pdf

Good Luck!
Ossama Ismail