CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

BIG DATA & ANALYTICS.

Lab03: On Developing a Benchmark Generator & Solver for Minesweeper.

BACKGROUND.

On Lab01 we developed a sequential-algorithm for solving the Minesweeper problem presented in the Irish Collegiate Programming Contest (IrlCPC) edition of 2017: http://multimedia.ucc.ie/Public/training/cycle2/IrlCPC-ProblemSet2017.pdf

On "Lectures 03-05: Distributed Programming" we explored the benchmark generator & solver app for running the problem of computing list inversions of person P vs an entire population.

LAB EXERCISE.

On Lab03 our goal is to adapt the benchmark generator & solver app from "Lectures 03-05: Distributed Programming" to generate and solve a benchmark for the Minesweeper problem. In particular:

- The benchmark generator will adapt the file **create_benchmark.py** in order to create k new files [file_1.txt, file_2.txt, ... , file_k.txt]. Each of these files represent a input for a Minesweeper game, following the format we saw in Lab01. In particular, to create each of these files, the following input parameters are needed:
  - num_rows => Rows of the Minesweeper board.
  - num_columns => Columns of the Minesweeper board.
  - percentage_mines => Percentage of positions in the board being a mine.
- The benchmark solver will adapt the **solve_benchmark.py** and **problem_algorithms.py** in order to solve the k files previously generated. In particular we envision 4 possible execution modes:
  - Execution Mode 1:
    - Solve the benchmark files in sequential order (using 1 single core).
    - Solve each file in sequential order (using 1 single core).
  - Execution Mode 2:

        - Solve the benchmark files in sequential order (using 1 single core).
        - Solve each file in distributed order (using more than 1 core).

    - Execution Mode 3:
        - Solve the benchmark files in distributed order (using more than 1 core).
        - Solve each file in sequential order (using 1 single core).
    - Execution Mode 4:
        - Solve the benchmark files in distributed order (using more than 1 core).
        - Solve each file in distributed order (using more than 1 core).

    Unfortunately, the execution mode 4 is not supported by the multi-threading library of Python 3, which triggers the following exception:
    AssertionError: daemonic processes are not allowed to have children.

For solving the benchmark files in distributed order, we can simply adapt the divide, map and reduce functions from **solve_benchmark.py** from "Lectures 03-05: Distributed Programming", dividing the files to be solved among the different cores.

For solving a single file in distributed order, we need to come up with novel divide, map and reduce functions allowing to split the problem into equally sized sub-parts, solve them in parallel and amalgamate their results, resp.

FOLDERS AND CODE FILES.

- The folder structure is equivalent to the one used in "Lectures 03-05: Distributed Programming", containing the folder **code**, **input_files** and **results**.
- Within the folder **code**:
    - The file **my_main.py** is provided complete. You do not need to adapt it.
    - The files **create_benchmark.py**, **solve_benchmark.py** and **problem_algorithms.py** have been directly taken from "Lectures 03-05: Distributed Programming". You have to adapt them to fulfil the new required functionality.