מחשוב מקבילי ומבוזר

תרגיל #2

The purpose of this exercise is to practice the MPI Cartesian Topology, Scatter, Gather and Sort

Write a parallel program to implement **Shearsort Algorithm** to sort a set of rectangles in ascending order.

Requirements:

- One of the processes reads all data from the text file **rectangles.dat**. Each line represents one rectangle – its id , width , height. The number of lines (rectangles) is equal to the number of processes launched.

- This process will display result of the sort and writes it to the file **result.dat**. It has to use **Scatter** to send data to processes, **Gather** to collect result of sorting.

- Use **Cartesian** Topology for communication between processors during sorting.

- Implement **Shear Sort** to sort the rectangles. The rectangles are sorted according to the value of its area. In case of equal areas, the rectangles are compared according to their id.

- Use **Odd Even Sort** to sort rows and columns. Assume that number of rows and columns is even. In other word, the number of processes (rectangles) can be represented as $n = (2k)^2$

- The input file **rectangles.dat** is organized as following:

```
0   6.1   6                    // Id, Height, Width
1   71.5   8
2   2.3   2
3   4.3    12.4
4   4   22.2
5   1.9   4   3
6   9.1   9
7   6   7
8   6.2   8.6
9   8   6
10   7   7
11   1.01  2.9
12   3   2
13   8.6   6.2
14   2.1   3.1   2.2
15   9   8   9
```

- The output file **result.dat** contains ids of rectangles according to sorted criteria

For example (does not correspond to the above input):

9 11 2 3 14 5 4 12 8 0 10 1 7 13 15 6

Note:

- You can suppose that number of processes is equal to the number of rectangles
- You need (2log(n)+1) row/column phases to sort n^2 rectangles.

Grading Policy:

- **10 points** for code quality:
  a. The code has to be divided into small functions (not more than 40 lines of code).
  b. Use meaningful names for variables, functions, files, constants.
  c. Place enough comments to understand the code
  d. No unused lines of code. Don't repeat the code – use functions!
  e. Write **README.TXT** file if special instructions are needed to run the solution. The file must be in the root folder of the project.

- **90 points** – for proper implementation of the requirements.

- The Homework must be delivered in time. No delay will be accepted.

Important:

- The homework may be performed in pairs. Only one member of pair submits the solution through the Moodle. The whole project must be zipped and named as

111111111_222222222.zip

Where **111111111** is ID of the one student and **222222222** is ID of another student

בהצלחה!