==========
Creating and solving random maze with graph algorithms on distributed memory
==========

Write C++ program with Message Passing Interface (MPI) extensions to programmatically create perfect maze of dimension 64x64 using the following two graph algorithms

1. BFS [3 marks]
2. Kruskal [5 marks]

A *perfect maze* is a maze with only one path between any two cells. Entry to the maze should be at top right (0,63) and exit from the maze should be at bottom left (63,0). Allowed moves from each cell are left, right, up and down to non-wall cells.

Solve the created perfect maze using the following two graph algorithms.

1. DFS [3 marks]
2. Dijkstra [5 marks]

==========
Your code should be organized inside src/ folder as follows

maze.cpp
generator/mazegenerator.hpp
generator/bfs.hpp
generator/kruskal.hpp
generator/mazegenerator.cpp
generator/bfs.cpp
generator/kruskal.cpp
solver/mazesolver.hpp
solver/dfs.hpp
solver/dijkstra.hpp
solver/mazesolver.cpp
solver/dfs.cpp
solver/dijkstra.cpp

We will run your program with 4 processes on a single machine with 4 processor cores, with $mpirun -np 4 ./maze.out -g [bfs/kruskal] -s [dfs/dijkstra]

Include Makefile with mpic++ command outside src/ folder to generate maze.out.

The output for each run should print from one processor the solved maze in command line with

(a) * for wall cells in the maze,
(b) space for non-wall cells in the maze not in solution path,
(c) P for non-wall cells in the maze in solution path,
(d) S for the entry cell
(e) E for the exit cell

Every run should generate a new random maze.

Include a PDF report explaining [4 marks]

(a) How you generate and solve random mazes using the graph algorithms
(b) How you take the 4 sequential graph algorithms and implement them in MPI for 4 processes
    (i) Discuss use of synchronization primitives for correctness (if any)
    (ii) MPI blocking vs. non-blocking calls used (if any)
    (iii) MPI reductions used (if any)
    (iv) Any optimizations done to handle sparsity of the maze graph, as each node in the a maze has constant number of neighbors in the graph
(c) Analyze analytically the speedup and efficiency of your MPI implementation over sequential (in order notation, not through actual measurement)

What to submit: entrynum1_entrynum2_entrynum3.zip containing

a. src/ folder as described above
b. Makefile
c. readme.md
d. report.pdf