Test generator

You are provided with a test generator in the file "*testgen.cpp*". The parameters of the generator can be found in the beginning of the code. A generator with the same structure is used to generate all the tests in the task.

The generator can generate tests of three different types by changing the "TestMode" constant between values 1, 2 or 3. The tests that will evaluate your solution will have equal number of tests from each type, but this does not necessarily apply to each subtask individually.

A description of the test generation process can be found below . It is recommended that participants do not spend too much time in details related to the process, but only to get acquainted with the general structure of the tests.

Algorithm for test generation

- The value of N is a random integer in the range [minN, maxN]
- The value of M is a random integer in the range [minM, maxM]
- The table is generated in one of the three ways depending on the "TestMode" parameter. Below you can read more detailed information about each way.
- The value of T is a random integer in the range [minT, maxT]
- The value of K is a random integer in the range [minK, maxK]
- Generate a random value minDist in the range [0, N+M]. This value is the same for each level and is used to generate the K robots as follows:
    1. Create an empty set of robot positions.
    2. Select a random cell that is at a Manhattan distance greater than or equal to minDist from each robot in the set.
        - If there is no such cell, then minDist is reduced by 1 **only for the current level**. Goes to step 1.
        - If there is such cell goes to step 3.
    3. A robot in the selected cell is added to the set
        - If the set contains K robots, we finish successfully.
        - If the set contains below K robots, we return to step 2.

Generating tables of type 1

- The value of **p** is a random integer in the range [30, 60]
- A random table is generated where each cell is blocked with probability **p%**
- The resulted table is divided into connected components from adjacent free cells.
- The components are being connected using the following algorithm:
    1. Breadth first search from the first component finds the shortest path to any other component, where "shortest" path means passing through a minimum number of blocked cells.
    2. All cells in the path found become free.
    3. The components are merged and the result is written as the first component.
    4. If there is more than one connected component we go back to step 1.

Generating tables of type 2/3

- An initial table is created:
    1. Initially, all cells are blocked.
    2. A random cell is chosen that becomes free.
    3. A blocked cell that has only one free adjacent cell is selected. If there is no such cell, we end.
        - For Type 2 tests, this cell is random among all eligible.
        - For Type 3 tests, a list of all cells meeting the condition is made. This list is sorted in increasing row number and secondary by increasing column number. If the list has up to 3 elements, the cell is selected randomly. If the list has more than 3 elements, the cell is selected randomly from the first-third of the list.
    4. The cell is now free and we go back to step 3
- (N+M) times this action is repeated:
    - We chose a random cell. If it is blocked we make it free.