Overview

You are to implement two classes that form part of a simple game, with robots moving around in a 2-dimensional space.

I have prescribed a particular external structure and names for your classes: please follow these precisely, because I will be testing your solutions automatically. You may also add private members to your classes, and any other functions you find useful.

Description

You should implement and test the following classes.

class robot

representing a robot moving around in a two-dimensional space.

Your class should have a default constructor that sets up a robot placed at the origin, and the following public methods: **void move north()** moves the robot one step to the north. **void move east()** moves the robot one step to the east. **void move south()** moves the robot one step to the south. **void move west()** moves the robot one step to the west.

int north() const returns the current distance north of the robot. (This could be negative, if the robot has moved south more often than north.)

int east() const returns the current distance east of the robot. (As with the previous function, this value might also be negative.)

int travelled() const returns the total distance travelled by this robot since it was created.

You should also define an external function (not a member function):

int distance(const robot &r) returns the distance of robot r from the origin according the *Manhattan metric*: the least number of steps it would take to move the robot back to the origin. The standard function abs(n), from the <cstdlib> standard header, may be useful here.

class game

A class holding many robots, each identified by a name (a string). No two robots in the same game will have the same name.

Your class should have a default constructor and public methods **int num robots() const** returns the total number of robots in the game (initially zero).

void move(const string &name, int dir) move the named robot one step in the specified direction (0 = north, 1 = east, 2 = south, 3 = west). If there is no robot of that name, one should be created at the origin and then moved as above.

int num within(int n) const returns the number of robots that are no more than n steps from the origin.

int max_travelled() const returns the furthest distance that any robot has travelled.

vector<string> robots by distance() const returns a collection of names of all the robots in the system, arranged in increasing order of distance from the origin.

You should use library algorithms where appropriate.

In order to implement move() efficiently, your class should hold the robots in a map.