This task is essentially corresponding to task 15.9 in the course book, with a slightly more detailed specification of the sensor model.

In short

You are assumed to work with robot localisation based on a Hidden Markov Model.

In detail

You are supposed to implement an HMM to do filtering for localisation in an environment with no landmarks. Consider the previously mentioned vacuum cleaner robot *in an empty room*, represented by an *n x m* rectangular grid. The robot's location is hidden; the only evidence available to you (the observer) is a noisy sensor that gives a direct, but vague, approximation to the robot's location. The sensor gives approximations *S = (x',y')* for the (true) location *L = (x, y),* the directly surrounding fields L_s or the "second surrounding ring" L_s2 according to the following.

The sensor reports
- the true location *L* with probability 0.1
- any of the n_Ls ∈ {3, 5, 8} existing surrounding fields *L_s* with probability 0.05 each
- any of the n_Ls2 ∈ {5, 6, 7, 9, 11, 16} existing "secondary" surrounding fields *L_s2* with probability 0.025 each
- "nothing" with probability 1.0 - 0.1 - n_Ls*0.05 - n_Ls2*0.025

This means that the sensor is more likely to produce "nothing" when the robot's true location is less than two steps from a wall or in a corner (there are also other possibilities of interpreting the sensor readings, but you should stick to this model for the task).

The robot moves according to the following strategy:

Pick random start heading h_0. For any new step pick new heading h_t+1 based on the current heading h_t according to:

P(h_t+1 = h_t | not encountering a wall) = 0.7
P(h_t+1 != h_t | not encountering a wall) = 0.3
P(h_t+1 = h_t | encountering a wall) = 0.0
P(h_t+1 != h_t | encountering a wall) = 1.0

It then moves in the direction h_t+1 by one step in the grid. This means essentially that it a) will always move one step and b) it can only move straight.

In case a new heading is to be found, the new one is randomly chosen from the possible ones (facing a wall somewhere along the wall leaves three, facing the wall in a corner leaves two options of where to turn).

Implement this as an HMM (according to the matrix-vector notation suggested in section 15.3.1 of the course book) and apply **forward filtering** to track the robot. This requires obviously a two-part implementation, as you need to simulate the robot and its movement (from which you also can simulate the sensor readings) to have some ground truth to evaluate your tracking against, and the HMM-based tracking algorithm as such. **Hint**: Encode the states such that each possible state represents

the position of the robot in the grid plus a heading. This means you will have rows * columns * 4 possible states.
Note the following: The filtering algorithm only knows sensor readings, which only represent positions in the grid, i.e., one sensor reading is equally likely for the four states that correspond to one position in the grid.
Hint 2: You should interpret a sensor reading of “nothing” according to the (known) model above - “nothing” will then mean that fields close to the walls and corners are slightly more likely to have caused the reading, but for center fields, the filtering step boils down to a pure prediction.

You can find a zip-file with a Java-based tool for visualisation of your transition model, your sensor model and your estimation (including some sort of user's guide) included in this archive! Feel free to use other programming languages (C++, Matlab, Python are ok as well) or tools, but if you choose Java, please make use of the visualisation - it presumably helps me a lot to understand your implementation!

Then: write a brief report on your approach, explaining your thoughts on the model and implementation, and, even more important, **discuss** the results and **answer** the following question:

How accurately can you track the robot’s path with this approach?

Hint 3: In terms of robot localisation it is often not relevant to know how often you are 100% correct with your estimate, but rather, how far "off" your estimate is from reality on average / how often. You could measure the distance between true location and estimate by using the manhattan distance (how many robot steps off), or the direct euclidean distance (looks nicer, but would not help a robot that can only move straight too much).

The report should roughly follow the outline of a scientific article, i.e., it should have an **introduction** explaining the problem to solve (self-contained and in your own words) and the general approach to be used, a section on your **method**, i.e., an explanation of how you interpreted the sensor model, how you implemented the transitions, assumptions you made, etc., a **result** section and finally a **discussion**, in which the above question should be answered. Additionally there should be one section on the **implementation**, where to find and how to run it. The report should then be submitted as a **PDF**-document!

(**IMPORTANT NOTES:** The path quoted must include your username /e.g. `~/TAI/a1/myprog` is NOT GOOD! Why?/ and you have to make sure it is accessible for the person evaluating your submission. Check the Linux `chmod` command for that purpose, please. The code should be accessible both for reading /i.e. source/ and running /i.e. executable/ without any additional demands, like compilation or usage of some particular IDE /read Eclipse/.) .

You may consult the code in the textbook code repository (or any other implementations), but the code you hand in must be primarily your work. You do not need to provide all or any code printout in the report - the code is available in your solution directory anyway - but only comments on its interesting parts.

The **deadline** for this assignment is Friday, 3rd of March, by which you MUST file in your working solution to the assignment (i.e. testing of your implementation according to your guide of usage should be possible).

Important remark: Remember that the time frame allotted to this assignment is approximately three days of your work, so please don't overdo it: a decent program will suffice to get a pass.

If you decide to use someone else's code (e.g., some library found on the web), please mention it both in the code and in your report: it is a matter of academic honesty. Lund University is committed to fighting every case of dishonesty or plagiarism.

The report should be sent to `tai@cs.lth.se` as an attachment to a mail message with the following subject line:
`Assignment 3 by username1 and username2`,
where `username` is your user name in the student computer system, like ada09jam.

The resulting programs should remain in your directory until you have been notified of the result, e.g. on the notice board and/or web or by e-mail. You may expect that your report and implementation will be examined within two weeks. If your report or implementation is unsatisfactory you will be given one chance to make the corrections and then to hand it in within a week after you have been notified (on the notice board and/or web or by e-mail).

Please send any requests for clarification to Elin_Anna.Topp@cs.lth.se or visit her personally at her office - preferably on Tuesdays between 9:30 and 11:30. In case she is not there when you attempt to meet her at other times you should try to make an appointment by e-mail.