CSCI-580 **Project 2** **Global Robot Localization problem**

In this project you will practice implementation of an algorithm for a robot localization problem.

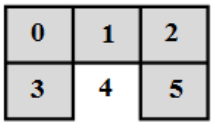
Given a grid world, a sensory error and a sequence of observations, a robot must estimate the room it is in at the time of the last observation.

A grid world is represented by an ***n*** x ***m*** matrix ***M*** of integers, each of which in binary shows that there are obstacles in directions <North, South, West, East> (in this order). In other words, M[i][j] = 10 means that at location [i][j] of the grid there are obstacles to the North and West of this location (since 10 in binary is 1010 and binary 1 represents an obstacle). A sequence of observations is given in capital letters: NW, NS, for example. NW means that there are obstacles to the North and West.

The matrix M is given in the input file "input.txt" and the sensory error together with a sequence of observations are given in a command line. For example, given the content of input.txt file is:

```
10  12  9
7   15  7
```

the corresponding grid world is the following:



In the given command line (*robot* is the executable program):

./*robot* input.txt 0.1 NW NS

0.1 is the sensory error and NW NS is the sequence of observations at time 1 and 2. If there are no obstacles in an observation, then input observation is 0 (character '0').

Your program must calculate estimation probabilities for each time and output the most likely state(s) where the robot is at a given time together with the maximum probability value. Please use ***long double*** variable to hold probabilities, and use directive <iomanip> and the following ***cout*** command to output the maximum estimation probability:

```
cout << std::showpoint << std::fixed  << setprecision(12) << estimation_probability ;
```

This will show decimal point and will output precisely 12 digits after the decimal point (so that everyone has the same output and your program will pass all tests on ***turnin*** system).

Output Format:

<estimation probability><space><state><space><state><space><endl>

For each observation, there should be one line of the output: first is the maximum estimation probability (in the format above), then all states that have the maximum estimation probability (states are numbered from smallest to largest, i.e. in increasing order such as 2 5 7) separated by spaces, and at the end of the line, use ***endl***. Please note, at the end of the last state, there is a space and then ***endl.***

Submission includes:

1. Since you use matrix multiplication, you need to write Matrix class that handles matrix multiplication and other operations on matrices (two files matrix.h and matrix.cpp need to be submitted).
2. Write another class Robot (two files robot.h and robot.cpp must be submitted) that implements global robot localization solution (calculates all the probabilities).

3. Write file main.cpp with the function main (that reads in the command line), in which you will create Robot object, read in input and implement the rest of the necessary functions/activities to complete the project.

4. The list of the files you need to submit to turnin: matrix.h, matrix.cpp, robot.h, robot.cpp, main.cpp.

5. Test your program on ***ecc-linux*** (host name is ***ecc-linux.csuchico.edu***) using the provided test files (the pairs of corresponding command-line-argument/output files such as t01.cmd and t01.out). Please name your executable program ***robot.*** To test your program, use the following commands on ecc-linux:

./robot $(cat tests/t01.cmd) > t01.my

diff t01.my tests/t01.out (if no output – your program's output matches the correct output)

vimdiff t01.my tests/t01.out (if output from *diff* command, use this one to see the difference between your output and correct one)

6. The instructor reserves the right to ask all or some students questions about their code to make sure that the code is their original work.

7. In addition to submitting files to turnin, you need to submit TAR file to Blackboard. Create a directory *csci581_proj1_UserName* (where *UserName* is your user name that you use for Blackboard), put all the files (matrix.h, matrix.cpp, robot.h, robot.cpp, main.cpp) in that directory, and run the following command inside the parent's directory of your *csci581_proj1_UserName* directory:
tar -cvf csci581_proj1_UserName.tar *csci581_proj1_UserName*

This will create the TAR file *csci581_proj1_UserName.tar* that you need to submit on Blackboard.

Failure to submit this file to Blackboard will result in 0pts for your project.

Grading rubric: Your program must to pass all tests (passing course requirement)

1. Grading will be done according to the number of passed tests (for example if there are 4 tests and only 2 passed by the first deadline, then the grade is 50/100 for the first submission, and the rest of tests will lose 15% and 30% for each late submission).

2. In addition, you need to break your code into classes (as required) and functions, so that your code in main is minimum necessary. Add comments that explain for each function what it does.

3. If you use any pointers, make sure you do not create memory leaks. If you pass large objects as parameters to a function, make sure to pass the objects by reference (or constant by reference if it is not changed by the function).

4. If a student were asked to answer the instructor's questions and is not able to answer some or all questions, the instructor will subtract 50-100% from his/her grade.

5. If a student failed to implement required Matrix and Robot classes inside the required files (and instead put all the functionality of the program inside main.cpp), then 50% of the total grade will be subtracted.