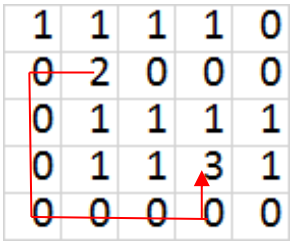
Academic honesty and plagiarism

- Plagiarism is unacceptable.
  - Code you submit must be your own. No copying, adapting, or submitting code you did not create is allowed. Working together and presenting variants of the same file is not acceptable. Here are some specific guidelines to make sure you don't cross the line:
    - Do not exchange programs or program fragments in any form on paper, via e-mail, photos, or by other means.
    - Do not copy solutions from any source, including the web or previous quarters' students.
    - Do not discuss code with other students at the level of detail that will lead to identical programs or program fragments.
  - Ask me if you are uncertain.
  - **All violations of the academic honesty will be immediately referred to the dean's office.**
- **Plagiarism detection will be applied to all code. It is very good and will catch you!! (20% of a class referred to dean!!)**

Lab 3: A star search

The task in this lab is to create a search algorithm to help a robot find its way to a destination using A star search and the Manhattan distance heuristic. The robot exists in a grid world named “map” of size MAP_WIDTH by MAP_HEIGHT, and it can move in all 4 directions (diagonals not allowed) through empty space cells only by steps of exactly 1 cell distance. The starting location of the robot is marked in map with a number “2” and the goal with a number “3”. The other two values you can find in the map are “1” for walls and “0” for empty space.

Sample map of size MAP_WIDTH 5 and MAP_HEIGHT 5



Robot starting location in map[1][1]
Goal in map[3][3]

You are to complete the code for the function: **astar_search**(map)
The A star search should use the Manhattan distance for its heuristic. The function should return a boolean value “true” if the destination was reached and “false” otherwise. An additional condition is to mark the map with a number “4” in all explored cells and with a number “5” in the cells that are part of path found.

To make sure everybody arrives to the same results (very important for the automated grader) when a tie (two nodes in the frontier have the same f(n) value) is found, first expand the node with the lowest x value, if there is still a tie, then expand the node with the lowest y value.

Considerations:

- We provided several maps to let you test your solution, but the grading will use a different set. Good performance on the test cases is not a guarantee you will do well on the grading cases.
- Remember to use the provided constants for the map boundaries and do not hardcode any values because during grading the dimensions of the map can be different.
- The running time of your algorithm cannot be longer than 10 seconds for a 15x15 maps or smaller, otherwise it will fail the grading tests.
- All maps will have a maximum of 1 possible path between the starting location and the destination (to make it easier).
- Unreachable goals are possible.
- There will be no loops in the maps (to make it easier).
- There will always be exactly 1 starting position and 1 goal.