CSE 2010, HW6
Due Tue Apr 21 at the start of your lab section; Submit
Server: class = cse2010, assignment = hw6SxIndividual
Due Tue Apr 21 at the end of your lab section; Submit
Server: class = cse2010, assignment = hw6SxGroupHelp

== Extra Credit 1 - 3 ==

Due Tue Apr 28 at the start of your lab section; Submit
Server: class = cse2010, assignment = hw6extraSxIndividual
Due Tue Apr 28 at the end of your lab section; Submit
Server: class = cse2010, assignment = hw6extraSxGroupHelp

x is 14, 23, j—section number or j for java submissions.

In the subgame of I/O Tower in the game of Tron, the player moves
Tron in a maze so that Tron can get to the I/O Tower quickly.
To prevent Tron from reaching his goal, grid bugs roam around trying to destroy Tron.
How would you design the bugs such that they can reach Tron quickly?

HW6 explores graph algorithms to simulate a simpler
version of the Tron game. Given a starting position in a 2D grid
world, a player’s goal is to move Tron to reach the I/O Tower
without running into a bug. At each step, the player can move
Tron up, down, left, or right to an adjacent empty cell.
Similarly, at each step, bugs can move in those four directions
to an adjacent cell that is empty or occupied by Tron.
For simplicity, the bugs move at the same speed as Tron.
The game ends when Tron reaches the tower or a bug reaches Tron.
The player moves Tron first, then each bug (in alphabetical
order) will move. Trying to reach Tron quickly, each bug
decides which direction to move based on the shortest path
from its cell to Tron’s cell. The distance from one cell to an
adjacent cell is 1. For easier debugging and testing, during
Breadth-First Search for the path, consider the valid adjacent
cells in this order: up, down, left, and right. Each cell can be
empty or can have Tron, a bug, or an obstacle.

HW6: one round of the first move from Tron and the first
move from the bugs.

HW6 Extra Credit 1 (via hw6extra1.c) [10 points]: multiple
rounds of moves to the end of the game.

HW6 Extra Credit 2 (via hw6Extra2.c) [30 points]: Smarter
bugs know that Tron is likely to use the shortest path to the
I/O Tower so that he can get there quickly. Hence, the bugs
would prefer their paths to cross Tron’s shortest path to the
I/O Tower. One approach is to increase the “distance”
between adjacent cells that are not on Tron’s shortest path.

To find the shortest path for a smarter bug, use Dijkstra’s
algorithm. One round of the first move from the player
and the first move from the bugs.

HW6 Extra Credit 3 (via hw6extra3.c) [10 points]: same as
Extra Credit 2, but multiple rounds of moves to the end of
the game.

Input: Command-line argument for hw6.c is a filename of
the 2D grid world—the first line has number of rows and
columns of the world, the following lines have the initial world
represented by these characters:

- T represents Tron
- I represents the I/O Tower
- a, b, c, d, ... represent bugs
- # represents a stationary obstacle
- a space represents empty

During the game, via the keyboard, the player can input u,
d, l, and r to indicate moving up, down, left, and right to
an adjacent cell. If the input is invalid (incorrect letter or
the adjacent cell is not empty), prompt the player to re-enter.

Output: Output goes to the standard output (screen):

1. the world with row numbers on the top and columns
   numbers on the left
2. Please enter your move [u(p), d(own), l(eft), or r(ight)]:
3. the world with row numbers on the top and columns
   numbers on the left
4. For each bug (in alphabetical order), display its move
   (u/d/l/r), length of its shortest path to Tron, and cells
   on the shortest path starting with the bug cell before
   the move and ends with Tron’s cell:
   Bug a: move shortestPathLength (row1.col1) (row2.col2) ...
   ...
   Bug b: move shortestPathLength (row1.col1) (row2.col2) ...

Row 0 is at the top and column 0 is on the left. Sample output
(with player keyboard input) is on the course website.

For extra credit, the program repeatedly displays the output
above and terminates after:

1. Tron reaches I/O Tower: the program displays the final
   world and “Tron reaches I/O Tower”, or
2. one of the bugs at Tron’s cell: the program displays the
   final world and “A bug is not hungry any more!”

Sample input files and output are on the course website.

Submission: Submit hw6.c that has the main method
and other program files. Submissions for Individual and
GroupHelp have the same guidelines as HW1.

For extra Credit 1, 2, and/or 3, submit hw6extra1.c,
hw6extra2.c, and/or hw6extra3.c that have/have the main
method and other program files. GroupHelp and late
submissions are not applicable.

Note the late penalty on the syllabus if you submit after the
due date and time as specified at the top of the assignment.