CSE 2010, HW6, Fall 2020

Due Thu Nov 12 at the start of your lab section; Submit Server: class = cse2010, assignment = hw6SxIndividual
Due Thu Nov 12 at the end of your lab section; Submit Server: class = cse2010, assignment = hw6SxGroupHelp
=== Extra Credit 1 - 3 ===
Due Tue Nov 24 at the start of your lab section; Submit Server: class = cse2010, assignment = hw6extraSxIndividual
Due Tue Nov 24 at the end of your lab section; Submit Server: class = cse2010, assignment = hw6extraSxGroupHelp
x is 12 or 34—your section number (or j for java submissions).

Florida Tech students have been playing the in-person Humans vs. Zombies game on campus for a number of years. How would you simulate it on the computer?

HW6 explores graph algorithms with a simpler computer simulation/game. Given a starting position in a 2D grid world, a player's goal is to reach the destination (e.g., home sweet home) and avoid stationary obstacles (e.g., buildings, garbage dumpster, and lamp posts (ouch!)) and undesirable moving creatures (e.g., zombies, alligators, bears (in Florida? yes), and barking dogs). The creatures are hungry and try to reach the player (breakfast/lunch/dinner?) as soon as possible. At each step, the player can move up, down, left, or right to an adjacent empty cell in the world. Similarly, at each step, the creatures can move in those four directions to an adjacent empty cell. Fortunately, the creatures move at the same speed as the player.

The player starts the first move, then all the creatures (in alphabetical order) will move. Each creature decides which direction to move based on the shortest path from its cell to the player'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 have a player, a creature, or an obstacle, or can be empty. The destination cell is empty. The player and all the creatures continue alternating moves. The game stops after the player reaches the destination (cell), or one of the creatures reaches the player (cell).

HW6 (via hw6.c): one round of the first move from the player and the first move from the creatures.

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

HW6 Extra Credit 2 (via hw6extra2.c) [30 points]: Smarter creatures know that the player is likely to use the shortest path to the destination so that he/she can get there quickly. Hence, to intercept the player, the creatures would prefer their paths to cross the player's shortest path to the destination. One approach is to increase the "distance" between adjacent cells that are not on the user's shortest path. For Extra Credit 2, the distance between two adjacent cells is:

- 1 if both cells are on the player's shortest path
- 2 if one of the two cells is on the player's shortest path
- 3 if both cells are not on the player's shortest path

To find the shortest path for a smarter creature, use Dijkstra's algorithm. One round of the first move from the player and the first move from the creatures.

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.

We will be evaluating your submission on code01.fit.edu; we recommend you to ensure that your submission runs on code01.fit.edu. To preserve invisible characters, we strongly recommend you to download and save, NOT copy and paste, input data files.

Input: Command-line argument for hw6.c is:

- filename of the 2D grid world—the first line has number of rows and columns (a maximum of 10 by 10) of the world, the following lines have the initial world represented by these characters:
  - a digit represents a player—0 in this assignment [single player for simpicity]
  - a letter [A-Z] represents a moving creature
  - # represents a stationary obstacle
  - * represents the destination, which is empty
  - 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 column numbers on the left
2. Player 0, please enter your move [u(p), d(own), l(elf), or r(ight)]:
3. the world with row numbers on the top and column numbers on the left
4. For each creature (in alphabetical order), display its move (u/d/l/r), length of its shortest path to the player, and cells on the shortest path starting with the creature cell before the move and ends with the player cell:
   Creature A: *move* *shortestPathLength* ($row1$,$col1$) ($row2$,$col2$) ...
   ...
   Creature Z: *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. the player arrives at the destination (cell), the program displays the final world and "Player 0 beats the hungry creatures!", or
2. one of the creatures arrives at the player (cell), the program displays the final world and "One creature is not hungry any more!"

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/has 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.