Adam Terhaerdt
01/25/2025
CS-380

Assignment 2

To start off, I want to say that I found a lot of enjoyment out of this programming assignment. I liked how the assignment did not hold your hand too much but gave me just enough information to figure it out on my own. I liked how the entire assignment built off one another, and you were using the functions you had just created to create new functions. Then finally, once I reached the `random` step, putting everything together was definitely a lot of fun (though there were a lot of bugs I had to fix). I genuinely enjoyed the final result and I find it fun to run the random program to see how often the program can randomly stumble its way onto the solution.

I found the majority of my bugs coming from accessing a 2D array. Getting my i's and j's mixed when accessing elements in the board, such as writing board[j][i] instead of board[i][j], was the cause of most of my bugs in the program. Also, the valid move function was more difficult than I first thought as I forgot to account for the goal state when writing this function. This led to some interesting behavior when writing my `random` function. I noticed that when I set n = 500, my program would either complete in the first >20 moves or take up the entire 500 moves. Statistically I thought this was weird but found that available moves were being calculated even though they should not have been.

As for the program itself, for the main function I used a match statement which would match the first argument given to the bash script to a certain function. Within each match statement, I wrote argument validation so I knew I was getting the correct amount of arguments from the command line. I defined constants such as wall, empty cell, goal, and also directions with x,y coordinates associated with each type of direction. Finally, I used global constants such as ROWS and COLS to keep track of the current gameboard's width and height. I thought having them global would be a problem for the `compare` function as I need to store two boards width and height, but for this function I just calculated the width and the height local to this function using the len() function.

For loading the board I would complete some string manipulation and format the board exactly how I wanted it. I would also throw specific errors if a file could not be found for certain IO errors. For printing the boards made sure to format the string by 2 for each cell. For getting a specific piece's moves I wrote some helper function to find the coordinates of the piece, then checked if it was a valid move by seeing if the new coordinates ran into a wall, were out of bounds, or if the space was not empty. Finally if it was a valid move I would append it to a list. For the normalization function I followed the algorithm that was given in the assignment and that worked perfectly. For applying the move, I used some *fancy* python syntax `[(x + dx, y + dy) *for* x, y *in* coords]` which would just calculate the new coordinates, then check if it was a valid move or not, if it was then switch the position with an empty cell.

Finally, putting it all together with the `random` function which used all of these functions plus the random library from python.

Results from testing routines:
Print:
Command #1:
`sh run.sh print ../SBP-levels/SBP-level0.txt `

Result #1:
5, 4
  1, -1, -1, 1, 1
  1, 0, 3, 4, 1
  1, 0, 2, 2, 1
  1, 1, 1, 1, 1

Done:
Command #2:
`sh run.sh done ../SBP-levels/SBP-level0-solved.txt `

Result #2:
True

Command #3:
`sh run.sh done ../SBP-levels/SBP-level0.txt`

Result #3:
False

Available Moves:
Command #4:
`sh run.sh availableMoves ../SBP-levels/SBP-level1.txt`

Result #4:
`
(3, DOWN)
(4, LEFT)
(6, LEFT)
`

Command #5:
`sh run.sh availableMoves ../SBP-levels/SBP-level0.txt`

Result #5:
```
(2, LEFT)
(3, LEFT)
```

Apply Moves:
Command #6:
`sh run.sh applyMove ../SBP-levels/SBP-level0.txt "(3, left)"`

Result #6:
```
5, 4
  1, -1, -1,  1,  1
  1,  3,  0,  4,  1
  1,  0,  2,  2,  1
  1,  1,  1,  1,  1
```

Comand #7:
`sh run.sh applyMove ../SBP-levels/SBP-level0.txt "(3, right)"`

Result #7:
Error: Invalid move

Compare:
Command #8:
`sh run.sh compare ../SBP-levels/SBP-level0.txt ../SBP-levels/SBP-level0.txt`

Result #8:
True

Command #9:
`sh run.sh compare ../SBP-levels/SBP-level0.txt ../SBP-levels/SBP-level0-test.txt`

Result #9:
False

Normal:
Command #10:
`sh run.sh norm ../SBP-levels/SBP-test-not-normalized.txt`

Result #10:
6, 8
1, 1, 1, 1, 1, 1
1, 3, 2, 2, 4, 1
1, 5, 2, 2, 6, 1
1, 7, 7, 8, 8, 1
1, 9, 9, 10, 10, 1
1, 0, 0, 0, 0, 1
1, 0, 0, 0, 0, 1
1, 1, -1, -1, 1, 1

Random:
Command #11:
`sh run.sh random ../SBP-levels/SBP-level0.txt 20`

Result #11:
`
Move 1: (3, LEFT)
Move 2: (2, LEFT)
Move 3: (2, RIGHT)
Move 4: (3, RIGHT)
Move 5: (3, LEFT)
Move 6: (4, LEFT)
Move 7: (3, DOWN)
Move 8: (3, RIGHT)
Move 9: (3, LEFT)
Move 10: (3, RIGHT)
Move 11: (4, UP)
Move 12: (2, LEFT)
Move 13: (4, DOWN)
Move 14: (4, UP)
Move 15: (4, DOWN)
Move 16: (3, RIGHT)
Move 17: (4, UP)
Move 18: (4, DOWN)

Move 19: (3, RIGHT)
Move 20: (2, UP)
`

Command #12:
`sh run.sh random ../SBP-levels/SBP-level0.txt 20`

Result #12:
Move 1: (3, LEFT)
Move 2: (2, LEFT)
Move 3: (3, RIGHT)
Move 4: (4, DOWN)
Move 5: (3, RIGHT)
Move 6: (2, UP)
Move 7: (2, DOWN)
Move 8: (2, UP)
Move 9: (2, DOWN)
Move 10: (2, UP)
Move 11: (2, DOWN)
Move 12: (2, UP)
Move 13: (4, LEFT)
Move 14: (4, LEFT)
Move 15: (2, UP)
Goal reached!

These are a few results from all of my procedures.