ITU Computer Engineering Department
BLG 223E Data Structures, Summer 2021
Homework #4
Due July 29, 2021 23:59

Definition

You are asked to implement a recursive rush hour puzzle board solver. Please pay attention to the following details in your implementations:

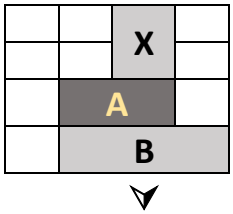
- Please read the material in the following link to get an idea about the puzzle game and how the game can be solved programmatically. PLEASE DO NOT ADOPT SOLUTION IDEAS FROM THE FOLLOWING LINK
    - https://www.michaelfogleman.com/rush/#GoVsC
- Your puzzle board is not going to contain any inner walls. It will only contain outer walls and an exit.
- With the definition of the homework you are provided with a model of the game over three C++ classes: Board, Move and Piece. These classes model the entities respective to their names. Please investigate the provided code throughly.
- You are free to modify the code that you've been provided with, in your solution.
- You are expected to develop a function (and any helper funcstions if necessary) that accepts a board and recursively checks if there's a solution.
- You are free to use global variables in your main file in your solution.
- Your solutions' efficiency is not the focus, developing a recursive solution is mandatory.
- Your program should read the board and piece inormation from a file, defined in the following section.
- Your solution is **NOT** expected to contain the smallest number of possible moves.

Input-Output

Your program should accept a filename as a command line parameter. In the accepted file, each line will contain a distinct command and parameters that should be processed by your program. In the input, the following commands may be given:

- Each file starts with an initial line containing three integers and a command string. First integer represents the dimensions of the board (NxN square), second and third integers represent the row and the column of the exit space. Input assumes initial cell starts by (0,0).
- The command string in the first line is either PRINT or CHECK. If it is PRINT, you should print the solution in some form (series of piece movements). Your programs would not be calico tested with PRINT commands (since solutions may vary) but manually tested to see if a plausible answer is produced.
- If the command string is CHECK you should only print if there is a solution to the board or not. (see calico file and inputs)
- Each line in the rest of the file represents a single piece. The line starts with a single character used as an ID fort he field. The next character is either H or V, designating if the piece is placed horizontal or vertical.
- The piece that would try to get out is always the first piece with ID X.
- Four integers follow the piece ID and orientation. These integers are couples are the initial and the final coordinates of the piece.

An example simple board and its representation as a file is provided below. The piece's coordinates are provided to follow its orientation. For instance if the piece's orientation is vertical, the first provided coordinate's row is always lower than the second coordinate. Exit cell is marked below with the arrow.




```
4 3 2 CHECK
X V 0 2 1 2
A H 2 1 2 2
B H 3 1 3 3
```

The input file would be error-free both syntactically and semantically.

Deliver

Please zip and deliver the directory structure defined below:

- HW4 : Topmost folder, that will contain all the folders in your submission. No other files should be present under this folder in your submission.
- HW4/src: Contains all the *.cpp files
- HW4/src/main.cpp: Contains your main function, recursive function(s) and other code you want to deliver.
- HW4/src/Board.cpp: Represents a board in a particular state.
- HW4/src/Piece.cpp: Represents a single piece in the board.
- HW4/src/Move.cpp: Represents a move performed by a piece.
- HW4/include: Contains all the header files you use
- HW4/bin: An empty directory that will contain objective files when your project is compiled

Please check the calico test file in the homework definition to see how your files will be compiled and tested.

Important: Specific to this homework, you have been provided with a single test case. You may generate your own cases by checking out the link in the homework definition (by omitting inner walls). Your homeworks will be tested with 5-7 different boards.

Restrictions and Guidelines

- Compilation environment: Only the code that can be compiled by the environment of the container definition provided in ninova will be accepted.
- Testing of your program will be performed using Calico (https://calico.readthedocs.io/en/latest/). Test cases that will be used to test your homework is provided as an attachment in ninova.
- **STL usage is allowed and promoted.**
- **This homework is for individual submissions**. Any kind of code sharing or code adaptation from an external source is strictly forbidden. Submitted code will undergo a plagiarism check process and plagiarsim penalties may be given if the submitted code's similarity is approved by the instuctor.
- Make sure you write your name and number in all of the files of your project, in the following format:
  /* @Author
  Student Name:<studentname>
  Student ID :<studentid>
  Date:<date>*/
- Only electronic submissions through Ninova will be accepted no later than deadline
- Use comments wherever necessary in your code to explain what you did.