Sheheryar Salman 01098645

2048 – EE1-07 Programming assignment 2016

1 OVERVIEW

I utilised a 2-D array for the foundation of my program. I proposed this since the grid size for this specific game is fixed, thus a static array would for example:

- Be memory efficient
- Prevent any input file streams from corrupting the game (if too many entries were put in then the program would only take the first 16)

The only drawback to this is that we cannot make custom grid sizes since variable length arrays are not approved by the standard C++ language. However as the specification did not require this, I chose arrays over vectors.

2 FUNCTIONS

There are a total of 12 void functions and 2 boolean functions. The approach I used featured a separation between the movement and the combination of identical elements. In our `play_game` function, the order of operations features movement, followed by combination of identical numbers, followed by another movement. The movement features a Boolean variable, `breakloop`, which governs the shifting of the number. Once an element is shifted, the `if` function (which requires the variable to be false) breaks and returns to the previous `for` loop which then completes the shifting for the rest of the elements.

The purpose of the third movement is to shift the combined number in the direction of the inputted move command, since as defined in the `combine` function, the element of the array that is behind is set to '0', meaning any elements before it will not have shifted as far as they should have. Followed by this series of functions, the '2' is generated by `spawnNumber` under the condition that the grid has a '0' present. This condition is very important and was something that caused problem in my program for a long time. Since I call the `spawnNumber` command in each user input cycle, if there were no '0' elements in the grid, then the program would come to a halt, as the function would not be able to execute itself. The addition of this condition meant that the '2' only prints if there is a free space, and does nothing if there are none.

The first of the two Boolean functions was there to confirm validity of the inputted move. The function creates a copy of the grid, which it then performs the move-combine-move operation on. If this copy is the same as the original, no change is detected, thus the move is invalid. This validity function is crucial, otherwise the program would print a '2' on an invalid move regardless.

The second Boolean function decides if the user loses the game. It is called after every user entry and checks if all moves are invalid. If so, no moves are possible, thus the game displays the "game over" message.

The final two void functions, `printGrid` and `initialiseGrid`, perform the printing and resetting of the grid. The resetting is for if the input filestream is not found, or if the user decides to restart the game at any point.

3 MAIN

The game continues due to a `while` loop in the main function. During gameplay, the only valid commands are *w, a, s, d* as well as *exit*. The user is able to also restart the game at any time. The only way to terminate the program is to exit the loop, which is possible only if the user enters *exit*, or if the user loses the game and chooses not to restart.