**Lab # 4**

Due: 2024-10-21

By: Omar Abdul, Anas Taimah, Yusuf Khan

300228700, 300228842, 300293842

Course: CSI3120

Group: 40

**Code explanation:**

**Step 1 Input Sudoku Grids:** The sudoku grid inputs are all premade and saved in different 2D lists.

**Helper function get\_column:** The function takes the grid and a column index and returns the values of that column. It maps a new array taking the element at the column index from every row in the grid.

**Helper function get\_subgrid:** The function takes the row and column index of the value and returns the values of the subgrid it is located in. It determines the quadrant by taking the row value (0, 1, 2 or 3) / 2 which gives 0 (when row is 0, 1) or 1 (when row is 2, 3). Then it is multiplied by 2 giving us a final new row value of 0 or 2 which determines if the value is in the left or right half of the grid. The same calculation is done for column which determines if the value is in the top or bottom half of the grid. The subgrid is then determined and returned in the form of [top left element, top right element, bottom left element, bottom right element].

**Step 2 Check if number is valid:** This function checks if a number can be placed at a certain position without violating the sudoku rules, if it does the function will return false. It starts by checking if the number is equal to any number in the given row, then the given column (using the get\_column helper function), and then the 2x2 sub grid (using the get\_subgrid helper function). The function returns true if none of these conditions are met.

**Step 3 Verify if the initial grid is valid:** This function takes a grid as input and defines a recursive function check\_all\_cells which takes the current row and column position. The base case if the row value is able equal to 4 the function ends and returns true. The case when the row is not equal to 4 but the column is equal to 4 means it reached the end of the row and then the function is called again using the next row and the column 0. In the case neither are equal to 4, if the number at the position isn’t 0, the function copies the grid and changes the value at the row and column to 0. Then the same value is attempted to be placed in the new grid using the step 2 is\_valid function. If it can be placed in the new grid then the step 2 function returns true. All values in the original grid are checked until the base case is reached or until a value returns false which means the sudoku grid isn’t initially valid.

**Step 4 Solve the grid:** This function uses the backtracking algorithm to solve the sudoku puzzle. First it finds the first empty element of the grid and attempts to place a number from 1 to 4. For each number, it checks if it goes against the sudoku rules. If the number is valid, the function recursively tries to solve the rest of the grid keeping that number in the position. If an error occurs in the grid with a later number, the function backtracks to try the next number. The function stops when the puzzle is solved and returns true or if there isn’t a way to solve it, it returns false.

**Step 5 Display the grid:** This function uses an iterator to print the entire grid once the puzzle is solved.

**Step 6 Handle edge cases:** This function checks if the grid input was invalid or if it does not have a solution and notifies the user. Otherwise, if it is solved, it calls the print function.

**Main:** Calls the solve function on each sudoku puzzle that was defined.

**Test cases:**

|  |  |
| --- | --- |
| A screenshot of a computer code  Description automatically generated | These match with the expected results |

**ChatGPT declaration:** Our group used ChatGPT throughout the lab to better understand the program application so that we are able to develop our answers. ChatGPT was used for assistance to learn and understand how it worked and not to generate any answers or results. All solutions were implemented in our own words and approach.