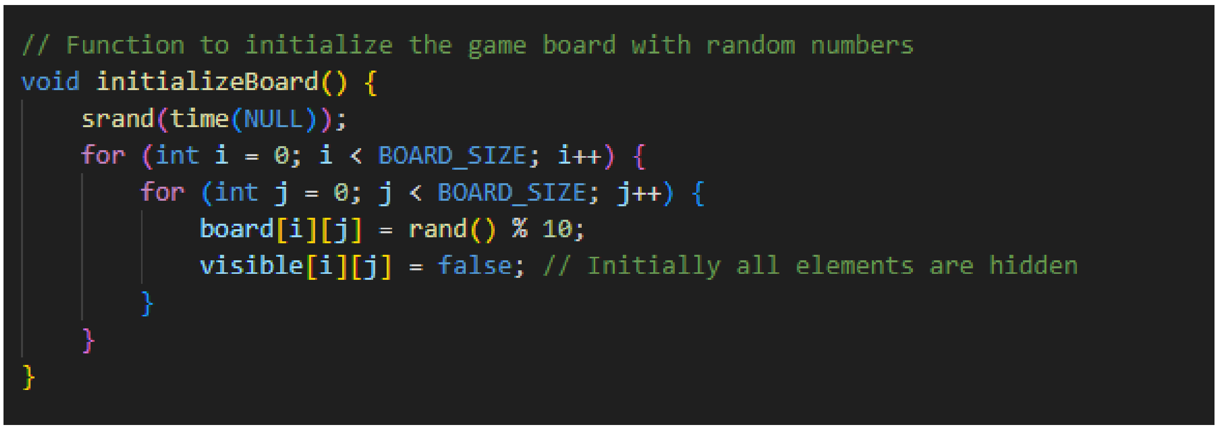
Assignment #2

Numbers Matching Game

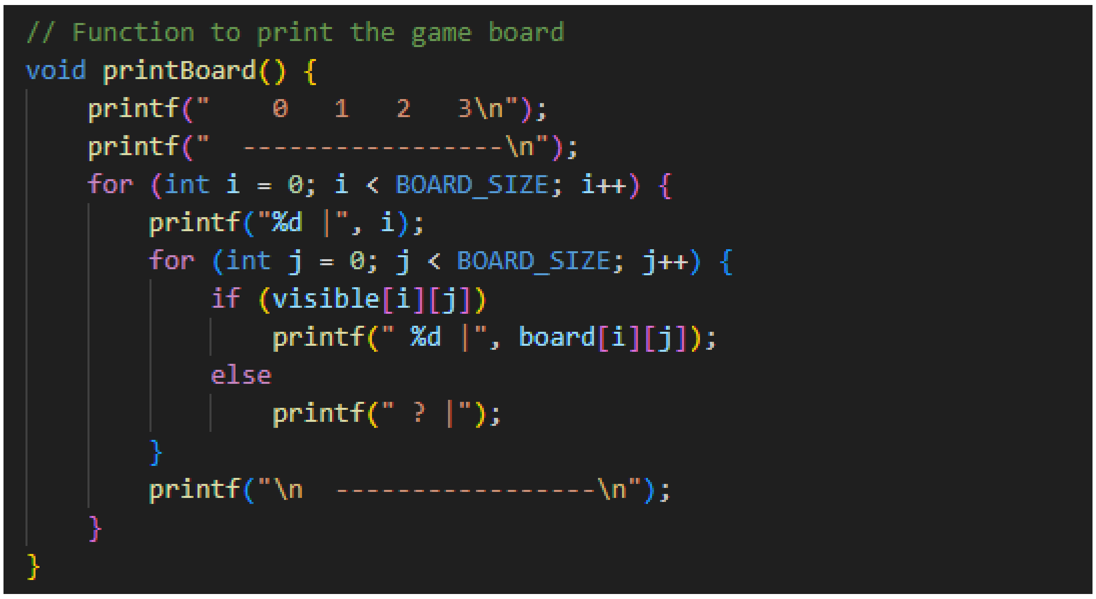
This Document Prepared by "Rabia YILDIRIM", "191805043"

1. (This step involves generating the game board with random numbers at the beginning of the game. The 'initializeBoard()' function is responsible for this task. It iterates through each cell of the board, assigning a random number to each cell and setting its visibility status to hidden initially. This ensures that every time the game starts, a new board with random numbers is generated, providing variability and challenge to the gameplay.



```c
// Function to initialize the game board with random numbers
void initializeBoard() {
    srand(time(NULL));
    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            board[i][j] = rand() % 10;
            visible[i][j] = false; // Initially all elements are hidden
        }
    }
}
```

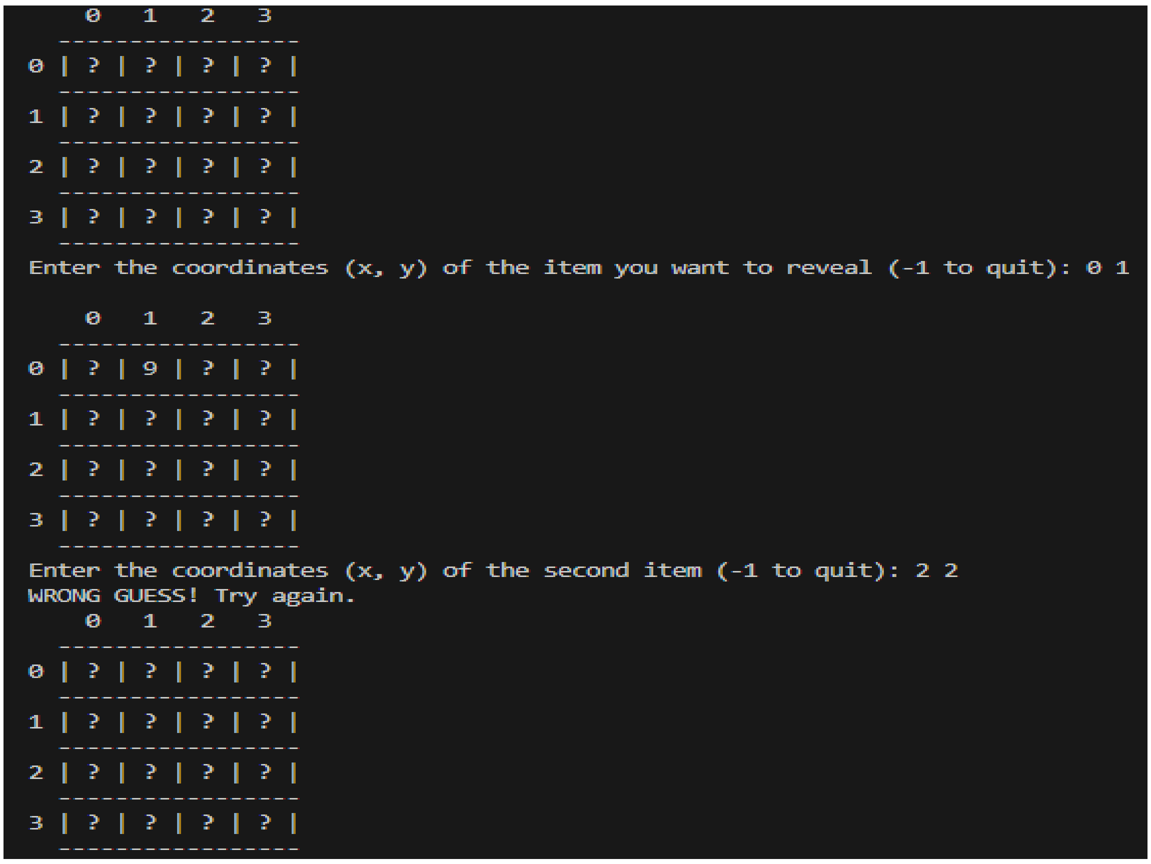
2. This step involves printing the initial game board at the beginning of the game. The 'printBoard()' function is responsible for this task. Each cell of the board is displayed with either the actual number (if visible) or a question mark **(?)** to indicate that the number is hidden. This creates a game board where all numbers are initially hidden, allowing the player to begin the game without knowing the positions of the numbers.



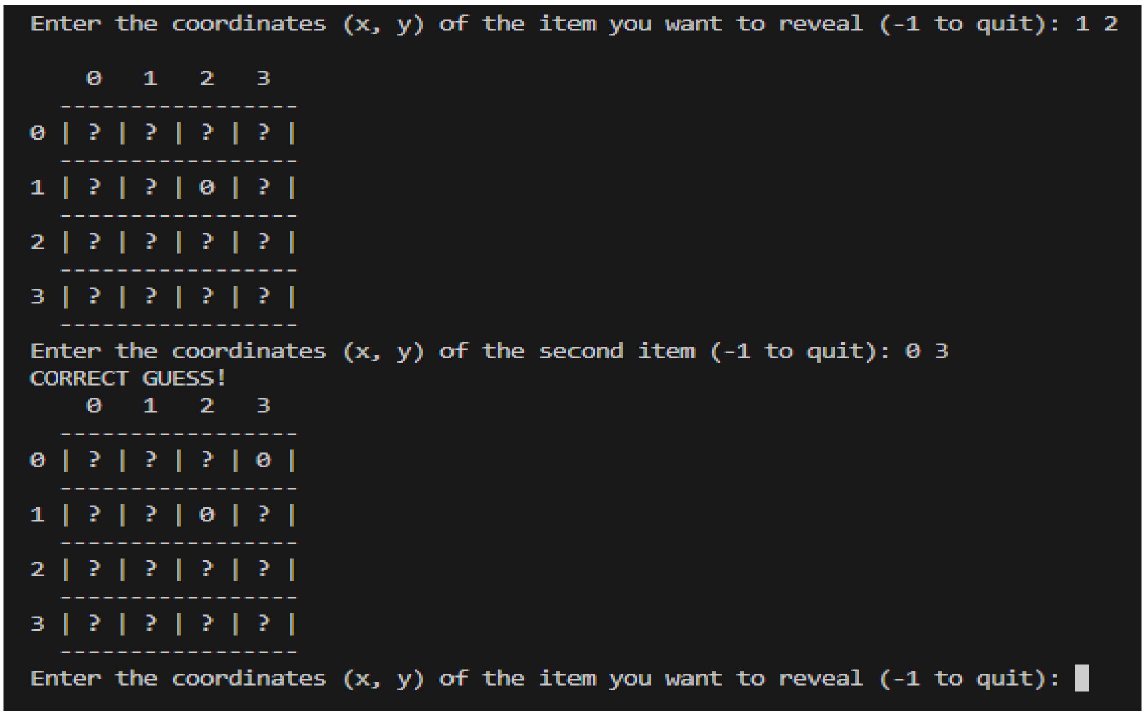
```c
// Function to print the game board
void printBoard() {
    printf("    0   1   2   3\n");
    printf("  ----------------\n");
    for (int i = 0; i < BOARD_SIZE; i++) {
        printf("%d |", i);
        for (int j = 0; j < BOARD_SIZE; j++) {
            if (visible[i][j])
                printf(" %d |", board[i][j]);
            else
                printf(" ? |");
        }
        printf("\n  ----------------\n");
    }
}
```

3. This console output represents a specific stage of the game. First, the question marks of all cells (?) is seen to be hidden with. Then, a loop begins, in which the player enters the coordinates and the board is updated. After each move, the updated board is printed on the screen. Finally, the greeting message is displayed when all the matching numbers are found.

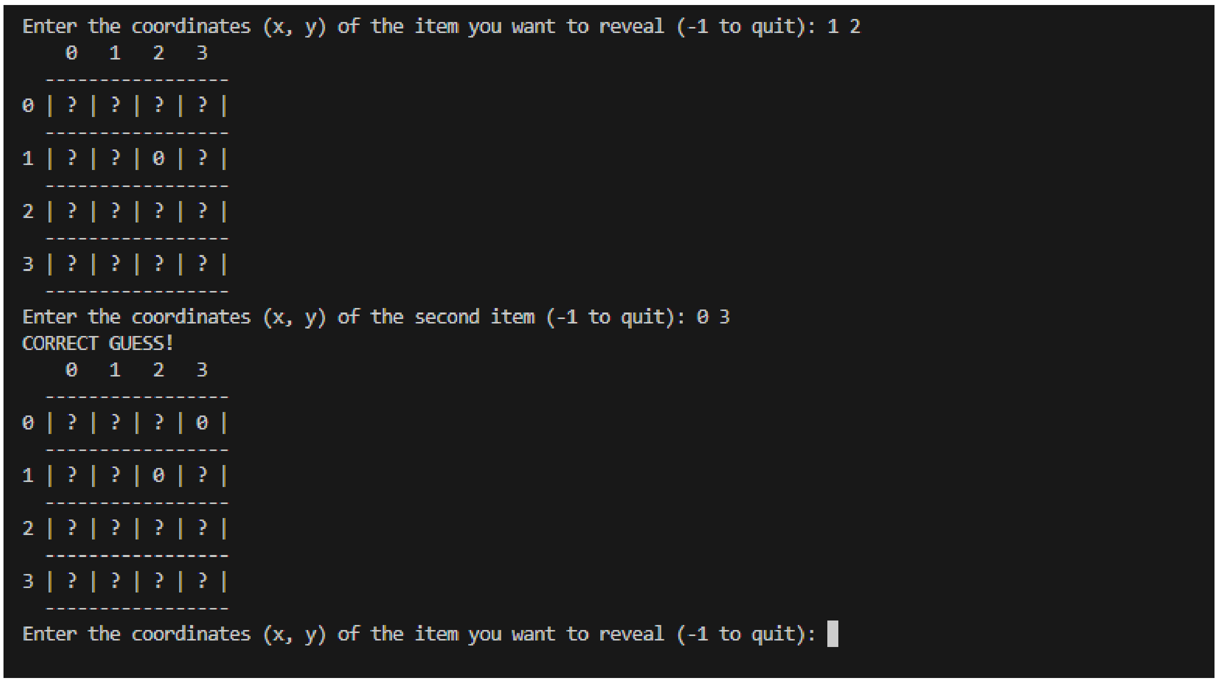
   The 'displayGameBoard()' function displays the game board and prints the updated board by calling the 'printBoard()' function within the game loop.



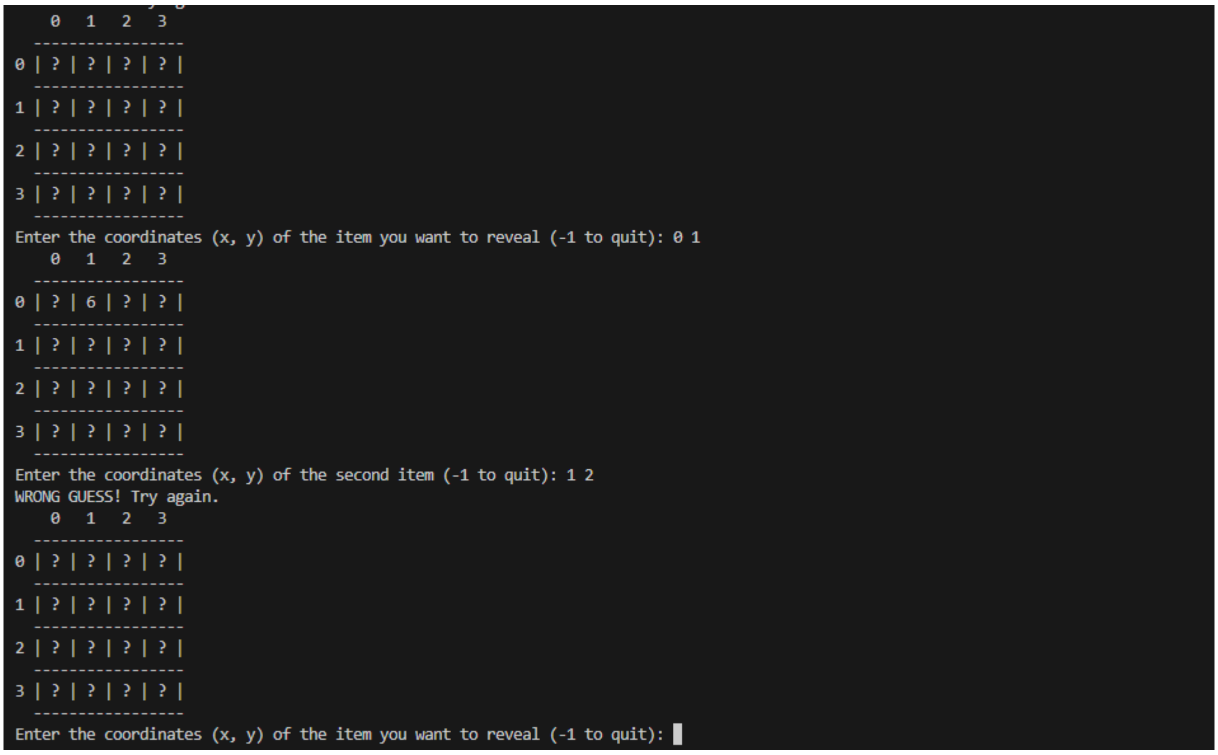
```
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 0 1

    0   1   2   3
  -----------------
0 | ? | 9 | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 2 2
WRONG GUESS! Try again.
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
```


```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 1 2

    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | 0 | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 0 3
CORRECT GUESS!
    0   1   2   3
  -----------------
0 | ? | ? | ? | 0 |
  -----------------
1 | ? | ? | 0 | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit):
```

4. This console output shows the situation where the two coordinates specified by the user match correctly. First, the user selects the first coordinate, and the cell with this coordinate becomes visible on the game board. Then, the user selects the second coordinate, and when the number in the cell with this coordinate matches the number in the cell with the first coordinate, the numbers in these two cells become visible. As a result, when the user makes a correct match, both numbers on the board become visible.

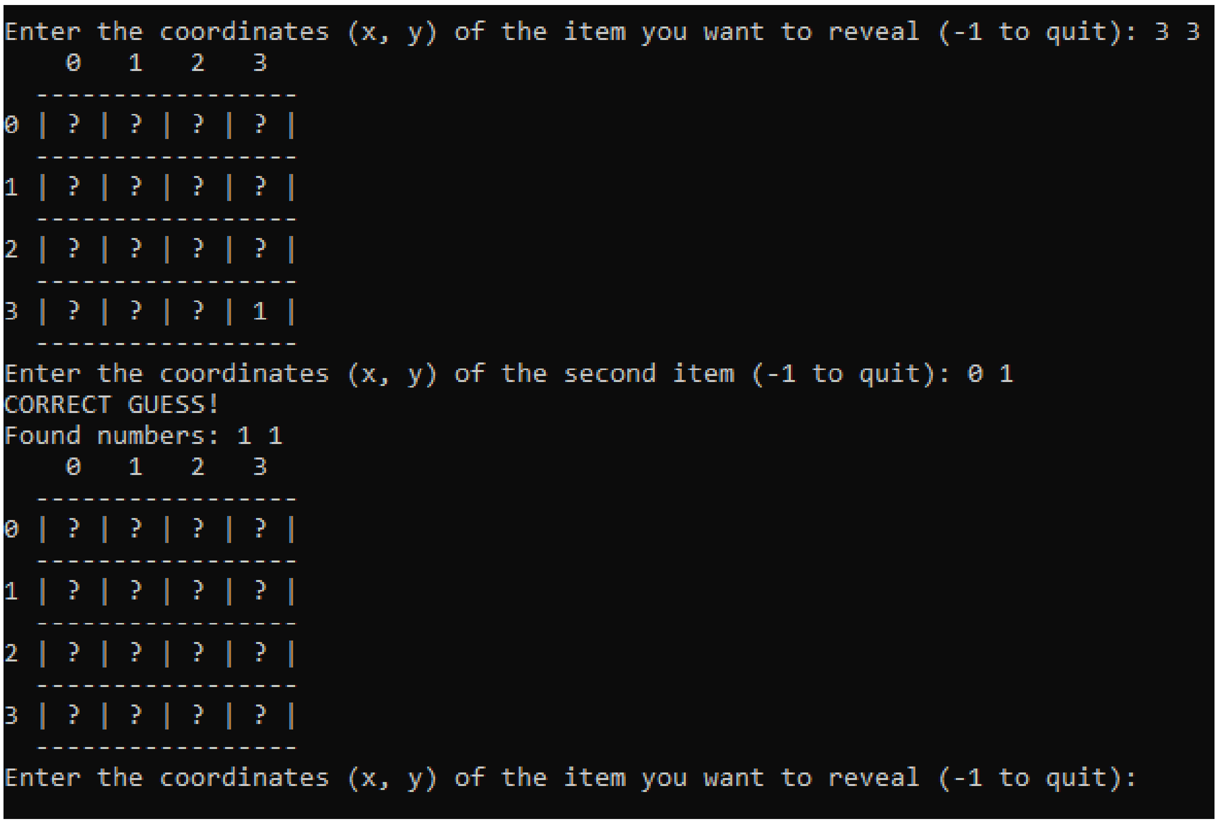

```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 1 2
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | 0 | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 0 3
CORRECT GUESS!
    0   1   2   3
  -----------------
0 | ? | ? | ? | 0 |
  -----------------
1 | ? | ? | 0 | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit):
```

5. İn this step, if the two elements selected by the user do not match, that is, if the values in the two predicted coordinates are not the same, we make these two elements invisible in the next iteration. In this way, when the user makes an incorrect guess, the elements that he made the wrong guess will not appear on the screen, and the player will not have to remember theseelements again when making his next move. This step improves the user experience by making the flow of the game easier and more consistent.



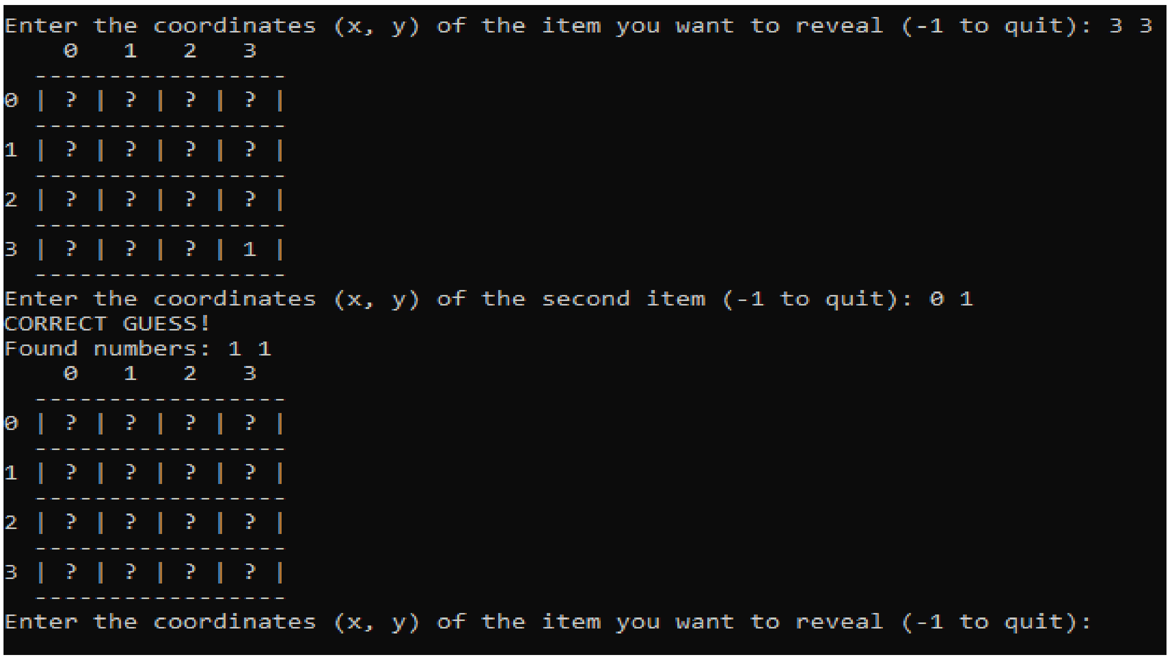
```
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 0 1
    0   1   2   3
  -----------------
0 | ? | 6 | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 1 2
WRONG GUESS! Try again.
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit):
```

6. In this step, it specifies that if two items match in the matching game, these items should be invisible in the next round. This ensures that the player will not see the items they have matched before ever again.



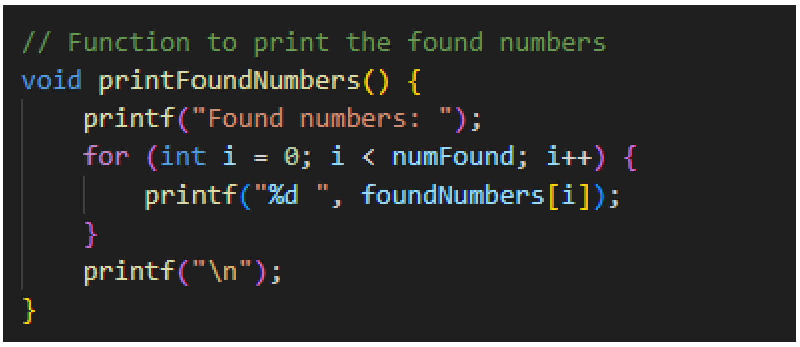
```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 3 3
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | 1 |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 0 1
CORRECT GUESS!
Found numbers: 1 1
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit):
```

7. In this step, it means that each number found in the matching game is added to the sequence of "found numbers" and this sequence is printed on the screen after each successful match. This allows the player to see which numbers they have found before and allows tracking of the numbers found as the game progresses.

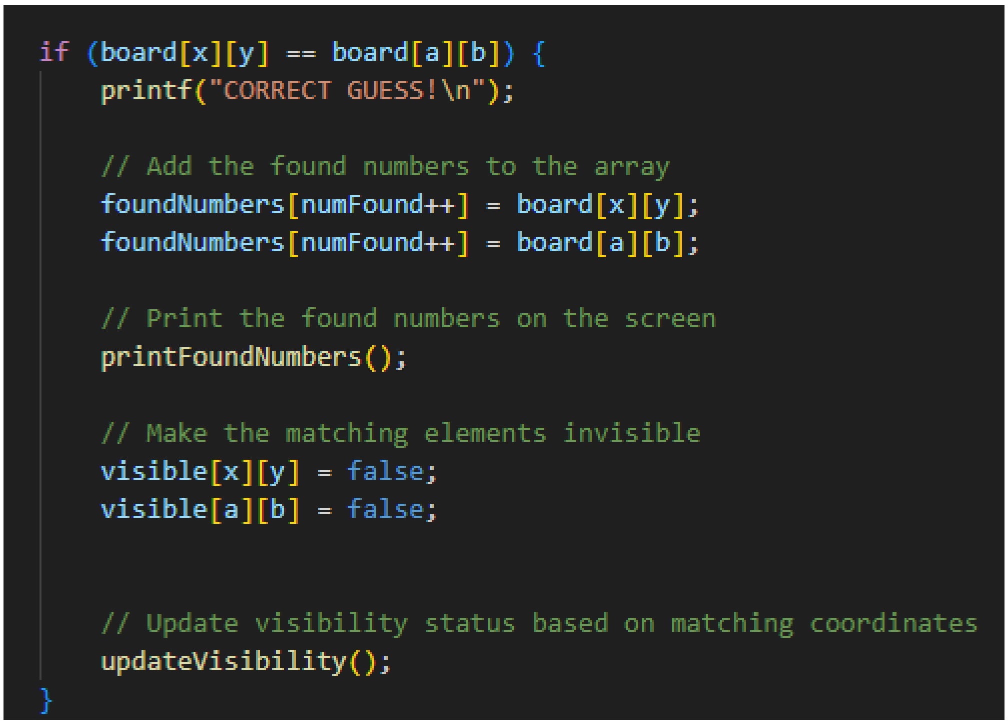


```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 3 3
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | 1 |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 0 1
CORRECT GUESS!
Found numbers: 1 1
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit):
```

The code defines a function called 'printFoundNumbers'. This function adds the found October numbers to an array and prints this array to the screen.

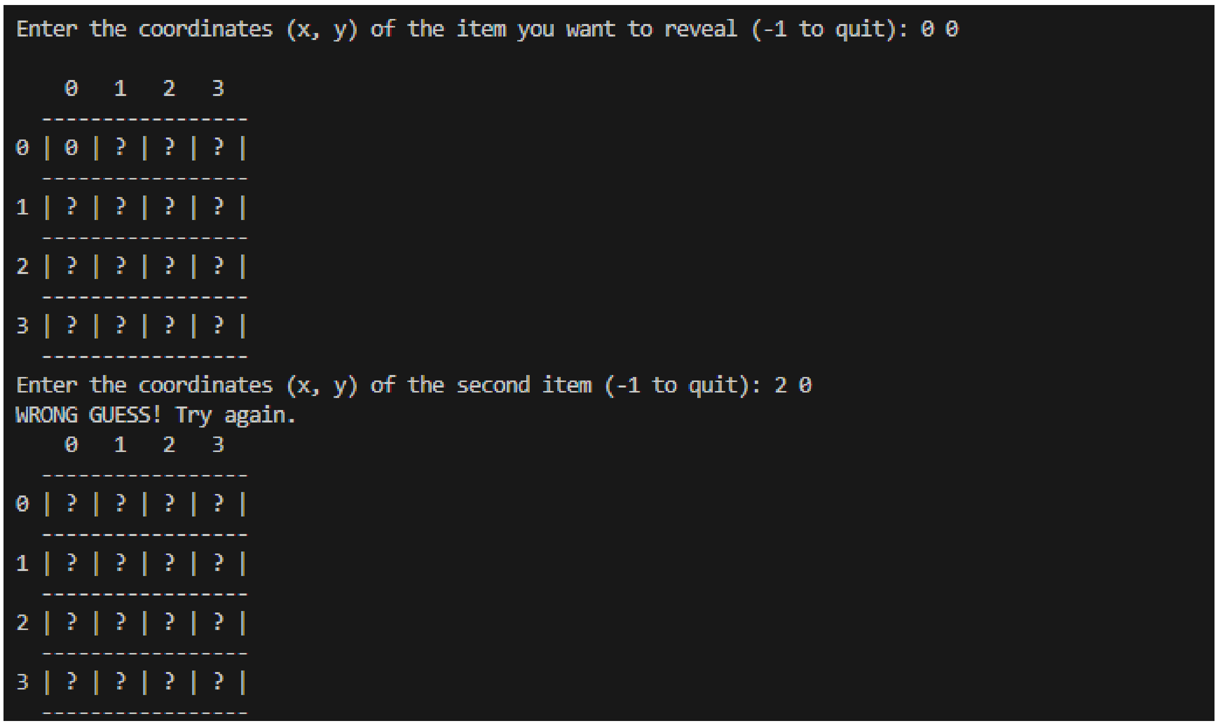


```c
// Function to print the found numbers
void printFoundNumbers() {
    printf("Found numbers: ");
    for (int i = 0; i < numFound; i++) {
        printf("%d ", foundNumbers[i]);
    }
    printf("\n");
}
```

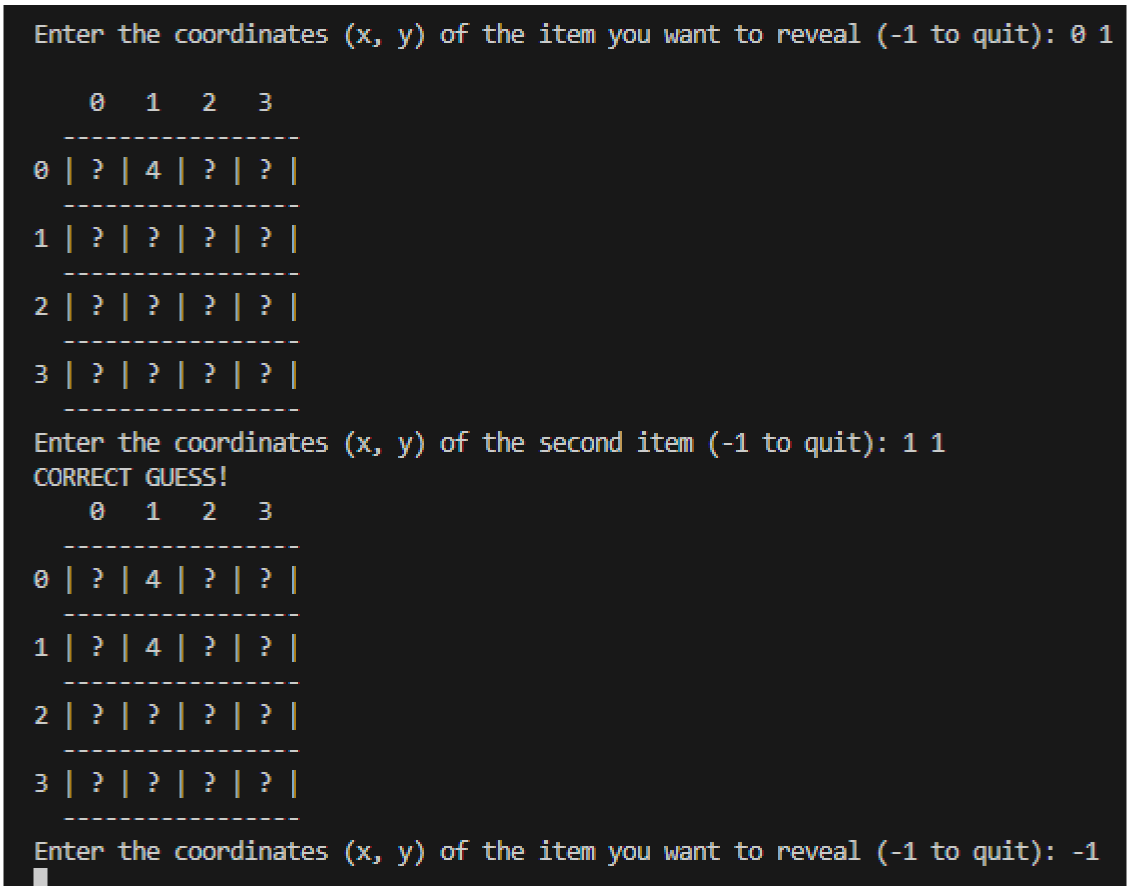


```c
if (board[x][y] == board[a][b]) {
    printf("CORRECT GUESS!\n");

    // Add the found numbers to the array
    foundNumbers[numFound++] = board[x][y];
    foundNumbers[numFound++] = board[a][b];

    // Print the found numbers on the screen
    printFoundNumbers();

    // Make the matching elements invisible
    visible[x][y] = false;
    visible[a][b] = false;


    // Update visibility status based on matching coordinates
    updateVisibility();
}
```

8. At this step, a warning should be given to the user to enter forecasts continuously to make new forecasts until the end of the game. The user should be encouraged to enter new estimates until he presses the -1 key or until all matches are found.

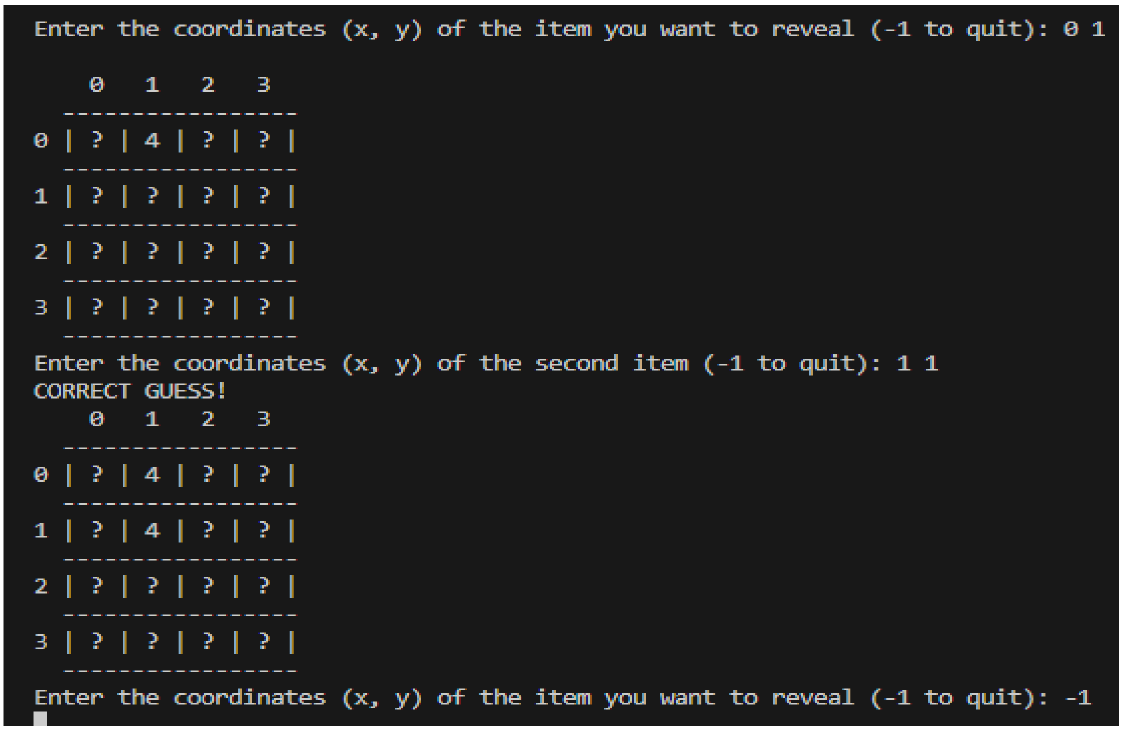


```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 0 0

    0   1   2   3
  -----------------
0 | 0 | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 2 0
WRONG GUESS! Try again.
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
```



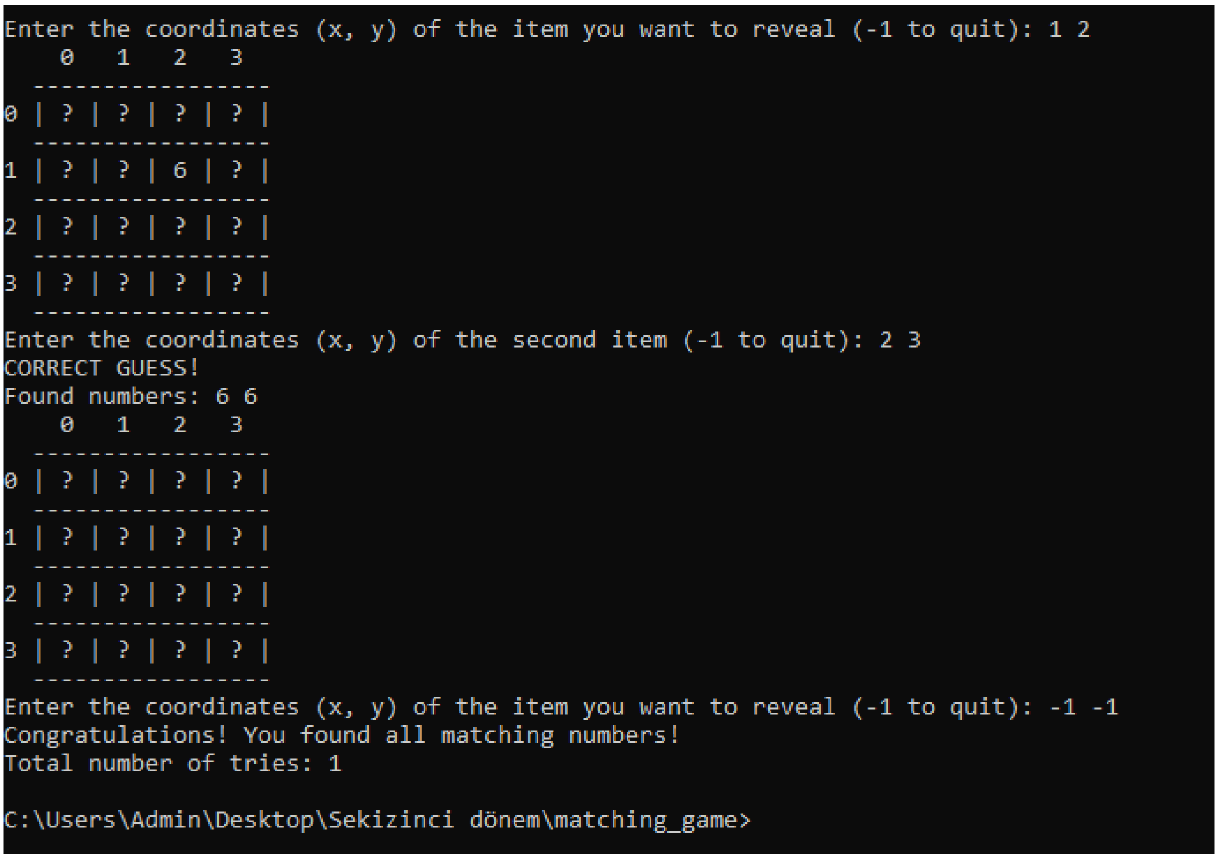
```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 0 1

    0   1   2   3
  -----------------
0 | ? | 4 | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 1 1
CORRECT GUESS!
    0   1   2   3
  -----------------
0 | ? | 4 | ? | ? |
  -----------------
1 | ? | 4 | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): -1
```

9. In this step, at each step, the user is given the option to continue playing the game or log out of the program. At each step, the user is prompted to enter a forecast, and then the user is asked if he wants to make a new forecast. The user can terminate the program by entering the "-1" key.



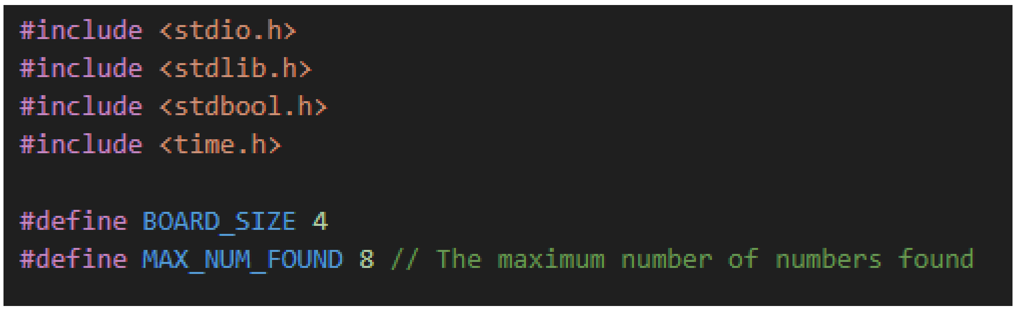
```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 0 1

    0   1   2   3
  -----------------
0 | ? | 4 | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 1 1
CORRECT GUESS!
    0   1   2   3
  -----------------
0 | ? | 4 | ? | ? |
  -----------------
1 | ? | 4 | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): -1
```

10. In this step, it seems that he wants to send a congratulatory message to the player and inform him of the total number of attempts if the game is completed. This message and information tells the player that he has successfully completed the game and how many attempts he has achieved after.



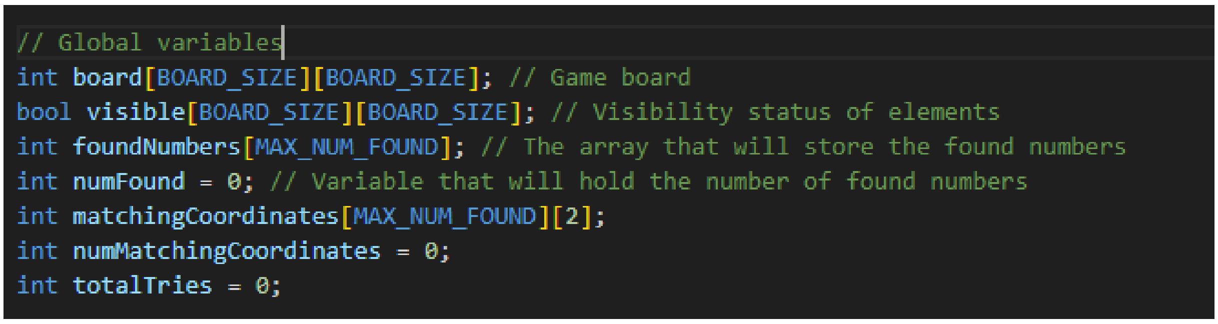
```
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): 1 2
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | 6 | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the second item (-1 to quit): 2 3
CORRECT GUESS!
Found numbers: 6 6
    0   1   2   3
  -----------------
0 | ? | ? | ? | ? |
  -----------------
1 | ? | ? | ? | ? |
  -----------------
2 | ? | ? | ? | ? |
  -----------------
3 | ? | ? | ? | ? |
  -----------------
Enter the coordinates (x, y) of the item you want to reveal (-1 to quit): -1 -1
Congratulations! You found all matching numbers!
Total number of tries: 1

C:\Users\Admin\Desktop\Sekizinci dönem\matching_game>
```

11.



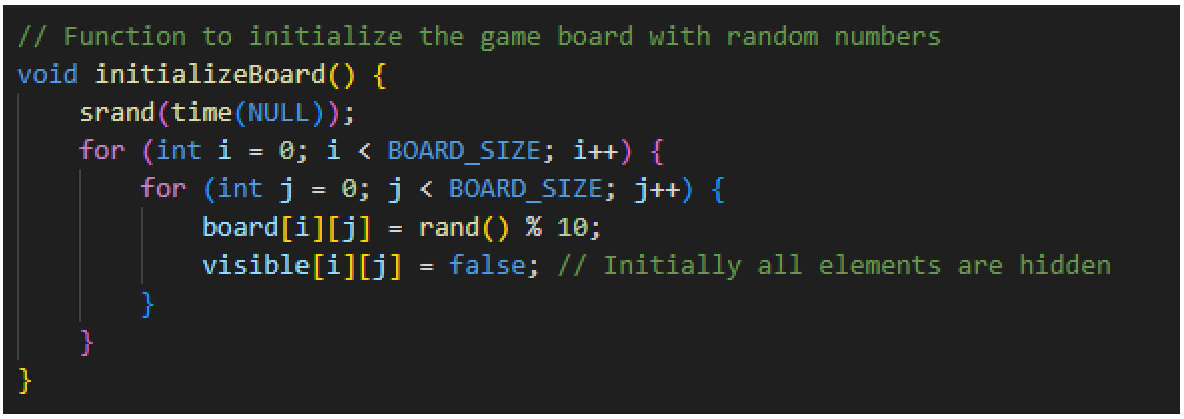
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

#define BOARD_SIZE 4
#define MAX_NUM_FOUND 8 // The maximum number of numbers found
```

In this section, the required header files (‘stdio.h’, ‘stdlib.h’, ‘stdbool.h’ and ‘time.h’) is included. Also, the ‘BOARD_SIZE’ and ‘MAX_NUM_FOUND’ constants are defined. ‘BOARD_SIZE’ specifies the size of the game board, and ‘MAX_NUM_FOUND’ specifies the maximum number of numbers found.



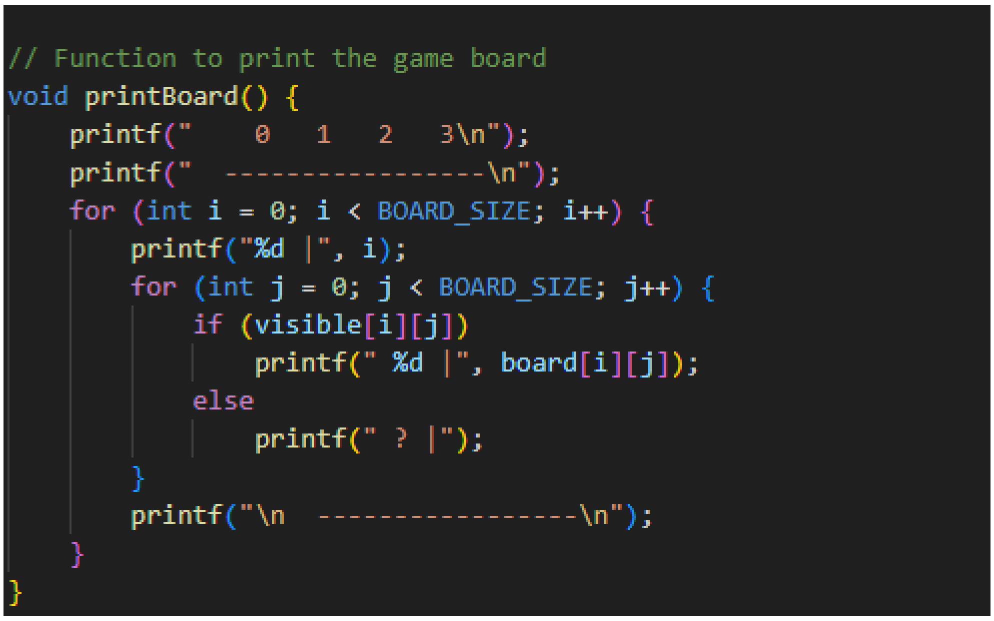
```c
// Global variables
int board[BOARD_SIZE][BOARD_SIZE]; // Game board
bool visible[BOARD_SIZE][BOARD_SIZE]; // Visibility status of elements
int foundNumbers[MAX_NUM_FOUND]; // The array that will store the found numbers
int numFound = 0; // Variable that will hold the number of found numbers
int matchingCoordinates[MAX_NUM_FOUND][2];
int numMatchingCoordinates = 0;
int totalTries = 0;
```

In this section, the global variables required for the game are defined. the ‘board’ keeps the game board, the visibility status of the cells on the ‘visible’ board, the numbers found in ‘foundNumbers’, the number of numbers found in ‘numFound’, the coordinates matching ‘matchingCoordinates’, and the number of coordinates matching ‘numMatchingCoordinates’. 'totalTries’ keeps the total number of attempts.



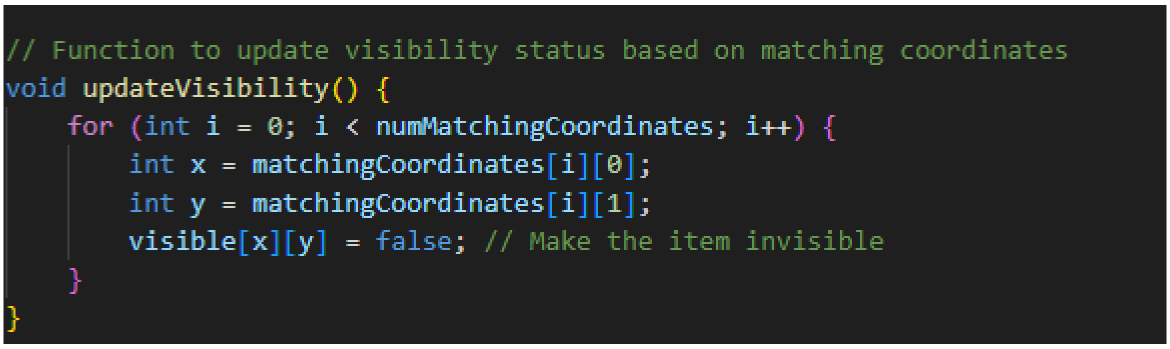
```c
// Function to initialize the game board with random numbers
void initializeBoard() {
    srand(time(NULL));
    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            board[i][j] = rand() % 10;
            visible[i][j] = false; // Initially all elements are hidden
        }
    }
}
```

The ‘initializeBoard()’ function fills the game board with random numbers. random number generation is started with 'srand(time(NULL))'and then a random number between 0 and 9 is assigned to each board cell. It is also noted that all cells are initially hidden.



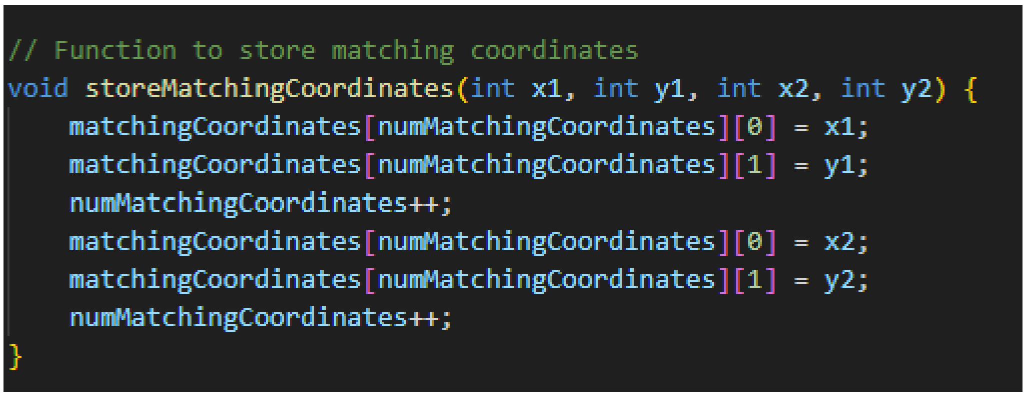
```c
// Function to print the game board
void printBoard() {
    printf("    0   1   2   3\n");
    printf("  -----------------\n");
    for (int i = 0; i < BOARD_SIZE; i++) {
        printf("%d |", i);
        for (int j = 0; j < BOARD_SIZE; j++) {
            if (visible[i][j])
                printf(" %d |", board[i][j]);
            else
                printf(" ? |");
        }
        printf("\n  -----------------\n");
    }
}
```

The ‘printBoard()’ function prints the game board to the console. The board cells can be visible or hidden. In visible cells, the value in the board cell, in hidden cells, the question mark (?) are printed.



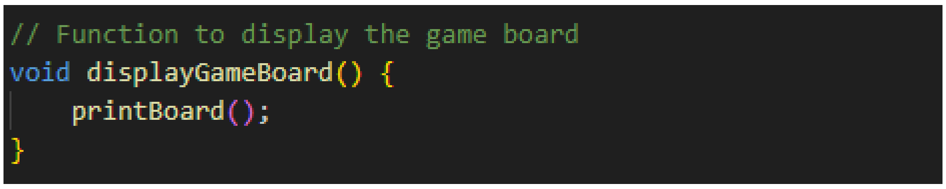
```c
// Function to update visibility status based on matching coordinates
void updateVisibility() {
    for (int i = 0; i < numMatchingCoordinates; i++) {
        int x = matchingCoordinates[i][0];
        int y = matchingCoordinates[i][1];
        visible[x][y] = false; // Make the item invisible
    }
}
```

The ‘updateVisibility()’ function updates the visibility of the matching coordinates. The visibility of the cells with matching coordinates is turned off (made false).



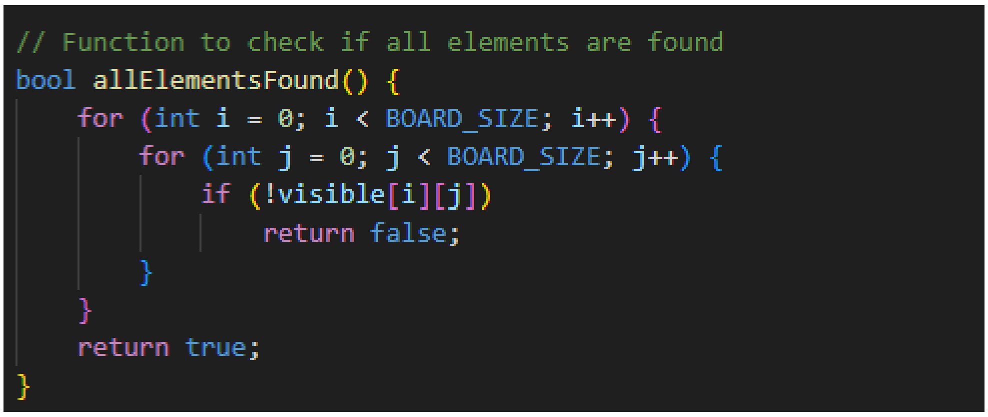
```c
// Function to store matching coordinates
void storeMatchingCoordinates(int x1, int y1, int x2, int y2) {
    matchingCoordinates[numMatchingCoordinates][0] = x1;
    matchingCoordinates[numMatchingCoordinates][1] = y1;
    numMatchingCoordinates++;
    matchingCoordinates[numMatchingCoordinates][0] = x2;
    matchingCoordinates[numMatchingCoordinates][1] = y2;
    numMatchingCoordinates++;
}
```

The ‘storeMatchingCoordinates()’ function stores the matching coordinates. The coordinates of both matching cells are transmitted to this function and added to the ‘matchingCoordinates’ array.



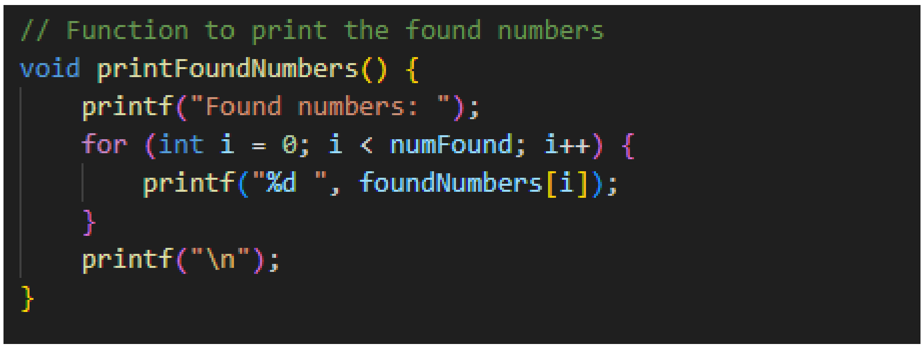
```c
// Function to display the game board
void displayGameBoard() {
    printBoard();
}
```

The ‘displayGameBoard()’ function prints the game board to the screen by calling the ‘printBoard()’ function.



```c
// Function to check if all elements are found
bool allElementsFound() {
    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            if (!visible[i][j])
                return false;
        }
    }
    return true;
}
```

The ‘allElementsFound()’ function checks whether all board cells are visible. If all cells appear, it returns true, otherwise it returns false.



```c
// Function to print the found numbers
void printFoundNumbers() {
    printf("Found numbers: ");
    for (int i = 0; i < numFound; i++) {
        printf("%d ", foundNumbers[i]);
    }
    printf("\n");
}
```

The ‘printFoundNumbers()’ function prints the found numbers on the screen. The numbers found in the ‘foundNumbers’ array are printed on the screen separated by spaces.

The ‘main()’ function provides the main operation of the game. It starts the game board, displays the board and gives instructions to the user to make moves. As the user makes moves, the coordinates entered by the user are checked and the board is updated. When a matching pair is found, a congratulatory message is sent to the user and the found numbers are printed. The total number of attempts is tracked, and when the game is completed, the user is given a congratulatory message along with the total number of attempts.