ASSIGNMENT 1

Project Brief:

The assignment is about practical application of the game of Draught using C language. All of the program's codes were done from the unique ideas of team members, and no external draught code has been referred or used while coding for this program.

Team Members:

Name	Registered number
Deepiga V	71762108007
Merlin Might V S	71762108027
Sakthi Abinaya S	71762108038
Samuela Abigail Mathew (Team Leader)	71762108039

Short Explanation of the Program:

First the program will ask to press ENTER key to start the game. screen_clear() function will clear the screen when ENTER key is pressed, and it'll often clear the screen according to wherever it is implemented in the program code. All text will be in different colors due to the color functions implemented in the program code. Then input of names of players will be taken, and scores of previous players will be displayed. Then game objectives and rules will be displayed.

The objective of the game is to capture your opponent's tokens. All tokens move only along the diagonal squares in forward direction, one square during each turn. A player may "push" one of his tokens onto one square occupied by his opponent's token, and this is called cutting of the opponent's token by the player. After the opponent's token is cut by the player's token, it is called the captured token, and it is removed off the board immediately.

The player's token is X and the opponent's token is O. Each have 12 tokens on a 8x8 square board, and turn 1 will be player 1's who'll be playing with token X. Player 2's token will be O, and player 2's turn is turn 2. Basically if turn number is divisible by 2, that turn is even turn and belongs to player 2. Odd turns belong to player 1.

The program will ask player to enter row number and column number of token to be moved and that of square where token is to be placed, and row number and column number will be displayed on all 4 sides of the board itself for the player's ease.

If the player makes an invalid move, an appropriate error message will be printed and the player will be given a 2^{nd} chance to make a valid move. The turn will go to the other player after the previous player's 2^{nd} chance gets over irrespective of whether the previous player made a valid move or an invalid move again.

If player's token reaches other end(opponent's side) of the board, 1 bonus turn will be given to that player and all of his tokens will become king during this bonus turn only. If a token becomes king, it

can move any number of squares in all directions. But if player gets a king during 2nd chance turn after making invalid move in first chance, bonus turn won't be given.

If player enters 9 9(surrender code) as row number and column number of token to be moved in his turn, he has quit the game and the other player will be declared as winner along with display of scores, and the program will end.

If player enters 8 8(help code) as row number and column number of token to be moved in his turn, a pop up help menu will be displayed, and his turn will continue until he makes a move. The help menu will display the game rules, and the player can use it in case he forgets the rules. But help menu can't be accessed if player presses 8 8 in 2nd chance turn or bonus turn.

The game will automatically end if all the tokens of the player is captured by the other player, and the other player will be declared as winner.

Identification of Variables, Constants, and Functions:

Variables/ Constants/ Functions	Identification
current_Player	It is a character data-type global array of size 25 which is used to store the name of the player who plays the current turn. In this program, it is indirectly used to determine which player name to display as the winner when the other player surrenders.

	Which player's name will be set as current_Player is determined by either move_O() function or move_X() function.
end_flag	It is an integer datatype global variable whose purpose is similar to Boolean variables. It is initialized to 0, and is used in end_game() function to determine whether to end the game or not. If end_flag is set to 1, then end_game() function will be called to end the program.
help_flag	It is an integer datatype global variable whose purpose is similar to Boolean variables. It is initialized to 0, and is used in help() function to determine whether to call help() function or not. If help_flag is set to 1, help() function will be called to display pop up help menu.
token	It is struct type global variable, and contains integer datatype variables named row and col
row	Used to represent y-coordinate of piece in checker board and move pieces accordingly
col	Used to represent x-coordinate of piece in checker board and

	move pieces accordingly
player	It is struct type global variable, and contains integer datatype variable named count, and character datatype array of size 25 called name.
count	Used for counting score of players and declaring the winner. Every time a player captures an opponent piece, count will get incremented by 1.
name	Used for storing names of players
player_X	It is a global variable of struct player type, and is used to represent player 1 to whom X pieces belong.
player_O	It is a global variable of struct player type, and is used to represent player 2 to whom O pieces belong.
checkers	It is a character datatype global 2-dimentional array having 8 rows and 8 columns. It contains (elements) pieces arranged like this- {'O',' ','O',' ','O',' ','O',' '}, {' ','O',' ','O',' ','O',' ','O'}, {'O',' ','O',' ','O',' ','O',' '}, {' ',' ',' ',' ',' ',' ',' ',' '},

	{' ',' ',' ',' ',' ',' ',' ',' ',' '}, {' ','X',' ','X',' ','X',' ','X'}, {'X',' ','X',' ','X',' ','X',' '}, {' ','X',' ','X',' ','X',' ','X'}
red(), yellow(), green(), blue(), purple(), cyan()	These are void type functions used to get colored text in red, yellow, green, blue, purple, and cyan color respectively.
screen_clear()	It is a void type function used for clearing screen using system("cls") in TurboC compilers. To clear screen in GCC/G++ compilers, use system("clear") instead.
old	It is a struct token type variable used to depict old row number and old column number of the token while playing draught.
new	It is a struct token type variable used to depict new row number and new column number of the token while playing draught.
end_game()	It is an integer type function that returns end_flag, and is used to end the game and display scores of players. It will be activated once end_flag is set to 1. If player enters 9 9 as row number and column number of token to be moved, the player has

	surrendered and other player will be declared as winner and program will be exited after display of players' scores. Or if either player captures all 12 opponent tokens, it'll end the game, display winner along with scores of the 2 players, and will exit the program.
draw_board()	It is a void type function used to print the 8x8 square board along with row number and column number on all 4 sides of the board. It also prints the checker pieces on squares of the board in correct positions using checkers[8][8] 2-D array
help()	It is an integer type function that returns help_flag, and it is executed once help_flag is set to 1. Basically, when player inputs 8 8 during his turn, a pop up help menu will be displayed and it'll allow the player to continue his move by giving 2nd chance to that player without passing the turn to next player.
move_X()	It is a void type function used for moving X pieces. It first calls check_valid_X() function to check if move made in 2nd chance is valid or not in case

	player 1 used 2nd chance. If check_flag_X is 0, it'll replace former position of checkers array with space character and replace new position in array with X. If piece O is already there in new position, X will replace it and count for player 1 will be incremented by 1. If check_flag_X=1, token won't be moved. And then it'll copy name of player 1 into current_Player, and it'll call draw_board() function to draw updated checker board for next turn if end_game() function returned 0.
move_O()	It is a void type function used for moving O pieces. It first calls check_valid_O() function to check if move made in 2nd chance is valid or not in case player 2 used 2nd chance. If check_flag_O is 0, it'll replace former position of checkers array with space character and replace new position in array with O. If piece X is already there in new position, O will replace it and count for player 2 will be incremented by 1. If check_flag_O=1, token won't be moved. And then it'll copy name of player 2 into current_Player, and it'll call draw_board()

	function to draw updated checker board for next turn if end_game() function returned 0.
get_turn()	It is an integer type function which returns a variable called rem, and rem is used to determine turn of players. If rem is zero, then it's player 2's turn. If rem is 1, it's player 1's turn. After that this function checks if end_flag is set to 1. If it's set to 1, then break statement is applied and end_game() function will be implemented. If rem is zero (even turn), it prints player 2's name and says O's turn. Then it gets row number and column number of token to be moved and that of square where token is to be placed from player 2. If 8 8 is pressed by player 2, help() function will be called , help_flag will be reset to 0, and 2nd chance to enter row number and column number will be given. After that, check_valid_O() function will be called to check if move made by player 2 is valid or not. If move made is invalid, check_flag_O is set to 1 and error message is printed by check_valid_O()

	function. And if end_flag is 0, check_flag_O will be reset to 0 and 2nd chance will be given. If end_flag is 1, end_game() function will be called. If end_flag is still 0, check_king_O() function will be called to check whether token O satisfies king condition or not, and if it satisfies, king_flag_O is set to 1 and then bonus turn will be given. Finally, move_O() function is called to move tokens accordingly and king_flag_O and check_flag_O are reset to 0. If rem is 1 (odd turn), it prints player 1's name and says X's turn. Then it gets row number and column number of token to be moved and that of square where token is to be placed from player 1. If 8 8 is pressed by player 1, help() function will be called, help_flag will be reset to 0 , and 2nd chance to enter row number and column number will be given. After that, check_valid_X() function will be called to check if move made by player 1 is valid or not. If move made is invalid, check_flag_X is set to 1 and error message is

	printed by check_valid_X() function. And if end_flag is 0, check_flag_X will be reset to 0 and 2nd chance will be given. If end_flag is 1, end_game() function will be called. If end_flag is still 0, check_king_X() function will be called to check whether token O satisfies king condition or not, and if it satisfies, king_flag_X is set to 1 and then bonus turn will be given. Finally, move_X() function is called to move tokens accordingly and king_flag_X and check_flag_X are reset to 0.
check_valid_X()	It is an integer type function which returns check_flag_X .This function is used to check whether the move made by player 1 abides with the game rules i.e. move made is valid or not. First it checks if player 1 entered 9 9 as row number and column number of token to be moved, and if it's true, it'll call end_game() function. If it's not true, it'll check if check_king_X() function is returning 0. If it returned 0, then it goes through conditions in for loops and if-else loops to see if the move made is invalid. If move made is found to be invalid,

	check_flag_X is set to 1 and it'll print appropriate error message. But if check_king_X() function is returning 1, it won't check most conditions to see if move made is invalid, although it will check a few conditions which must be applied in the game irrespective of whether it satisfies king condition or not.
check_flag_X	It is an integer datatype global variable whose purpose is similar to Boolean variables. It is initialized to 0, and is used in check_valid_X() function to determine whether the move made by player 1 is valid or not. If move made is invalid, check_flag_X is set to 1 and check_valid_X() function will be called to print error message.
check_valid_O()	It is an integer type function which returns check_flag_O. This function is used to check whether the move made by player 2 abides with the game rules i.e. move made is valid or not. First it checks if player 2 entered 9 9 as row number and column number of token to be moved, and if it's true, it'll call end_game() function. If it's not true, it'll check if check_king_O()

	function is returning 0. If it returned 0, then it goes through conditions in for loops and if-else loops to see if the move made is invalid. If move made is found to be invalid, check_flag_O is set to 1 and it'll print appropriate error message. But if check_king_O() function is returning 1, it won't check most conditions to see if move made is invalid, although it will check a few conditions which must be applied in the game irrespective of whether it satisfies king condition or not.
check_flag_O	It is an integer datatype global variable whose purpose is similar to Boolean variables. It is initialized to 0, and is used in check_valid_O() function to determine whether the move made by player 2 is valid or not. If move made is invalid, check_flag_O is set to 1 and check_valid_O() function will be called to print error message.
turn	It is an integer type variable used to get turn of the player and start the game. Value of this variable is determined by another variable called rem in get_turn() function.

rem	It is an integer type variable inside get_turn() function, and is initialized to zero. Once move_X() function is called inside get_turn() function, it'll be set to 1. Basically, rem is the remainder when variable called turn is divided by 2. If rem is 0, it is even turn i.e player 2's turn to play. If rem is 1, it is odd turn i.e player 1's turn. Since output of rem is either 0 or 1, it is similar to Boolean type variables.
store_prev_score()	It is a void type function used to open scoreboard.txt file and store names and scores of player 1 and player 2 in that file.
scoreboard.txt	It is a .txt type file in which scores of previous players are stored by store_prev_score() function. While displaying scores of previous players, this file is opened and read in main() function.
check_king_X()	It is an integer type function which returns a variable called king_flag_X ,and this function is used to check whether X tokens can be turned into king or not. If any X token reaches other end of the board (row 0), king_flag_X

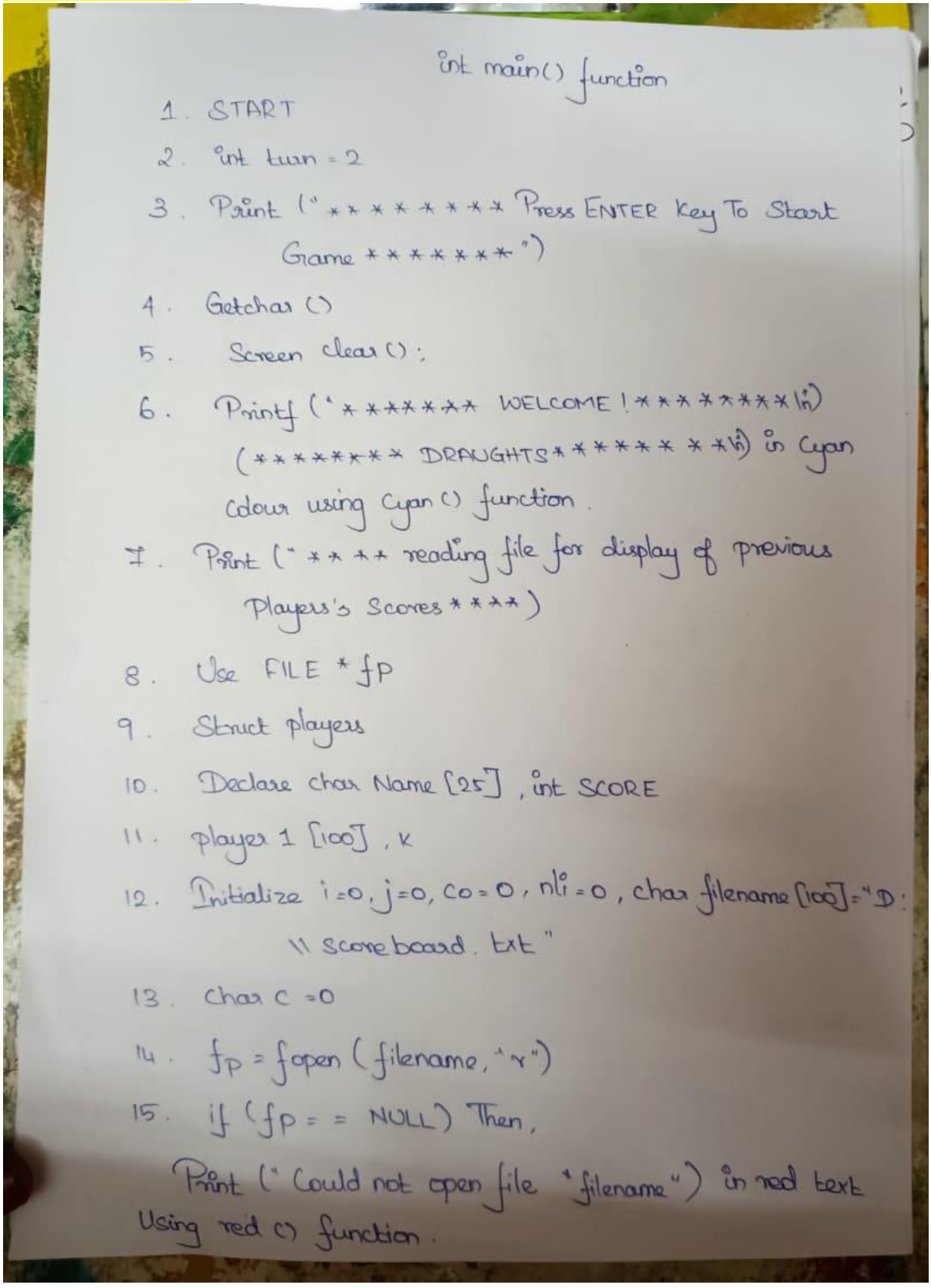
	is set 1.
check_king_O()	It is an integer type function which returns a variable called king_flag_O ,and this function is used to check whether O tokens can be turned into king or not. If any O token reaches other end of the board (row 7),king_flag_O is set 1.
king_flag_X	It is an integer datatype global variable whose purpose is similar to Boolean variables. It is initialized to 0, and is used in check_king_X() function to determine whether to activate king for X tokens or not. If any X token reaches row 0, king_flag_X is set to 1 and check_king_X() function will be activated.
king_flag_O	It is an integer datatype global variable whose purpose is similar to Boolean variables. It is initialized to 0, and is used in check_king_O() function to determine whether to activate king for O tokens or not. If any O token reaches row 7, king_flag_O is set to 1 and check_king_O() function will be activated.

Peer Assessment:

Name	Task done
Deepiga V	Getting player names input and display of rules using printf in main() function, draw_board() function
Merlin Might V S	store_prev_score() function, reading of the file storing previous players' scores in main() function
Sakthi Abinaya S	red() function, yellow() function, green() function, blue() function, purple() function, cyan() function
Samuela Abigail Mathew	Compiling all codes into single program, screen_clear() function, struct token, struct player, end_game() function, get_turn() function, move_X() function, move_O() function, check_valid_X() function, check_valid_O() function, help() function, check_king_X() function, check_king_O() function

Pseudo Code Algorithm:

Main function



int main() function

1. START
2. int turn = 2
3. Print ("******** Press ENTER Key To Start Game *******")
4. Getchar ()
5. Screen clear ();
6. Printf ("******* WELCOME ! ********\n) (******** DRAUGHTS ******** \n) in Cyan colour using Cyan () function.
7. Print ("**** reading file for display of previous Players's Scores ****)
8. Use FILE *fp
9. Struct players
10. Declare char Name [25], int SCORE
11. player 1 [100], k
12. Initialize i=0, j=0, co=0, nli=0, char filename [100]="D:\\ scoreboard. txt"
13. Char c =0
14. fp = fopen (filename, "r")
15. If (fp == NULL) Then,

Print ("Could not open file "filename") in red text Using red () function.

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

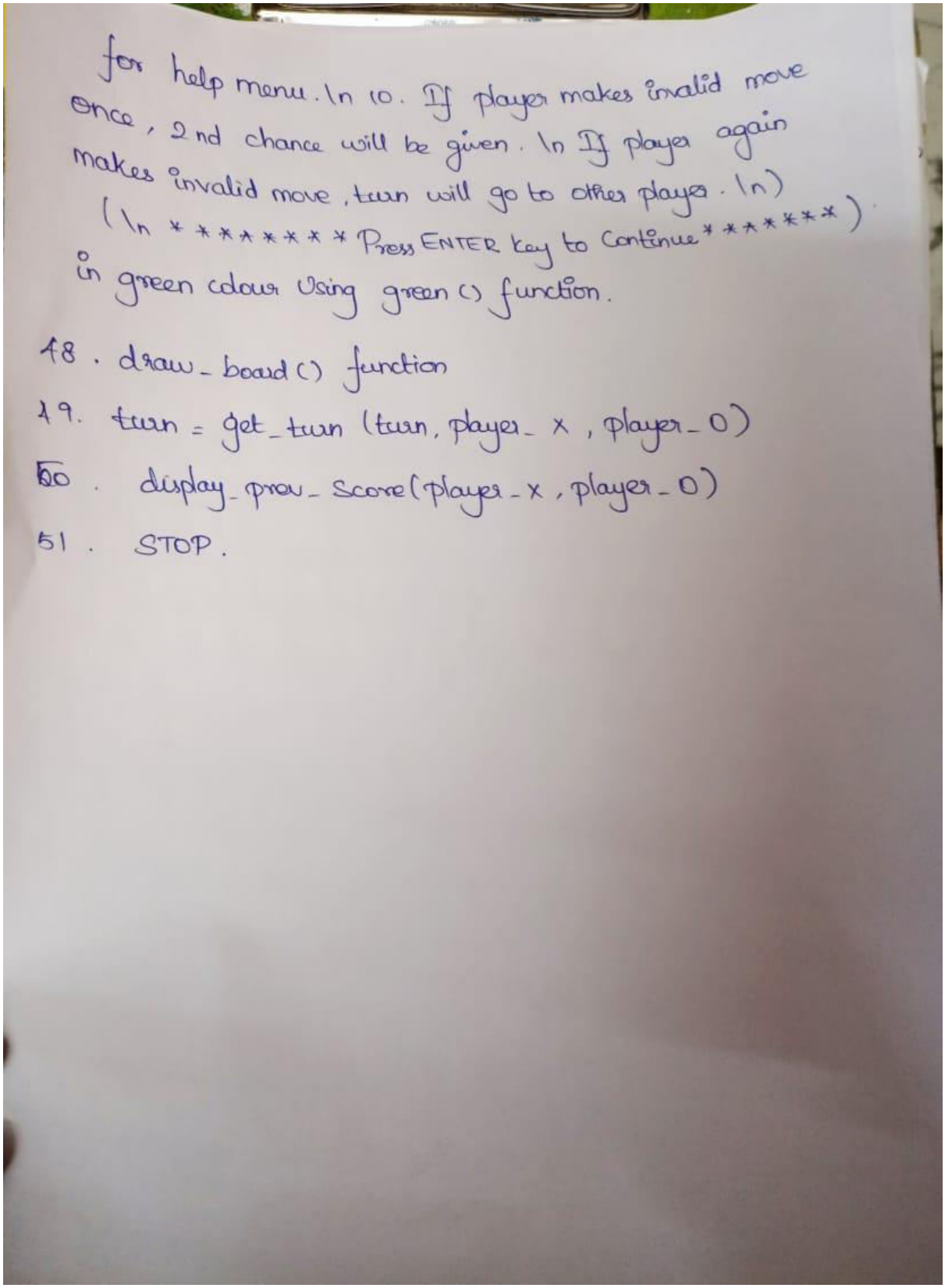
```
for (c = getc(fp); c != EOF; c = getc(fp))
If (c == '\n') Then,
    nli = nli + 1
    fclose (fp)
FILE * Scoreboardfile 1 = fopen ("D:\\Scoreboard.txt","r");
Print (" *READ SCORE FROM TEXTFILE *")
for i value =0 till i is less than nli
fscanf (Scoreboardfile 1, display player 1 [i], Name,)
                                          SCORE
for i value =0 till i is less than nli
for j value =0 till j is less than nli - 1 then;
if (player 1 [j]. Score < player 1 [j+1]. SCORE) then
        k = player [j]
    player 1 [j] = player 1 [j+1];
    Player 1 [j+1] = k
fclose (Scoreboardfile 1)
print ("Previous players Score")
Print ("RANK \t Name \t \t Score")
Print ("----------")
Print ("TOP 3 Players")
For i value =0 till i is less than 10
```

31. if (i == 3) then
32. Print ("_ _ _ _ _ _ _ _ * * * * * * * * - - - - - - - -")
33. Print (i+1 Value \t \t \t Player1 [i].NAME Value , \t \t Player1 [i].score value")
34. Print ("**** END of reading file for display of players")
35. Print (" Enter name of player 1")
36. Input player_x.name
37. Print (" Enter name of player 2")
38. Input (player_0.name)
39. Print (*** Good Day ***) Using red() function in red colour text.
40. Print (" player_x.name Value ") in green colour text using green() function.
41. Print ("and") in red colour text using red() function.
42. Print (" player_0.name Value") in green colour text Using green() function.
43. Print ("let's start the Game. All the best !!! ***") in red colour text using red() function.
44. Print (" objective : Opponent token's Capture") Using blue() function.
45. Print (" # cutting of opponent's token : A player may 1* Push *1 one of his tokens onto one square occupied

by his opponent's token Using blue() function.

16. Print ("# Captured token : The opponent's token is cut by the player's token and removed from the board) (\n# points : Number of tokens cut by the player is the player's score.\n) (\n***** Press ENTER key to Continue ******\n) in blue colour Using blue() function.

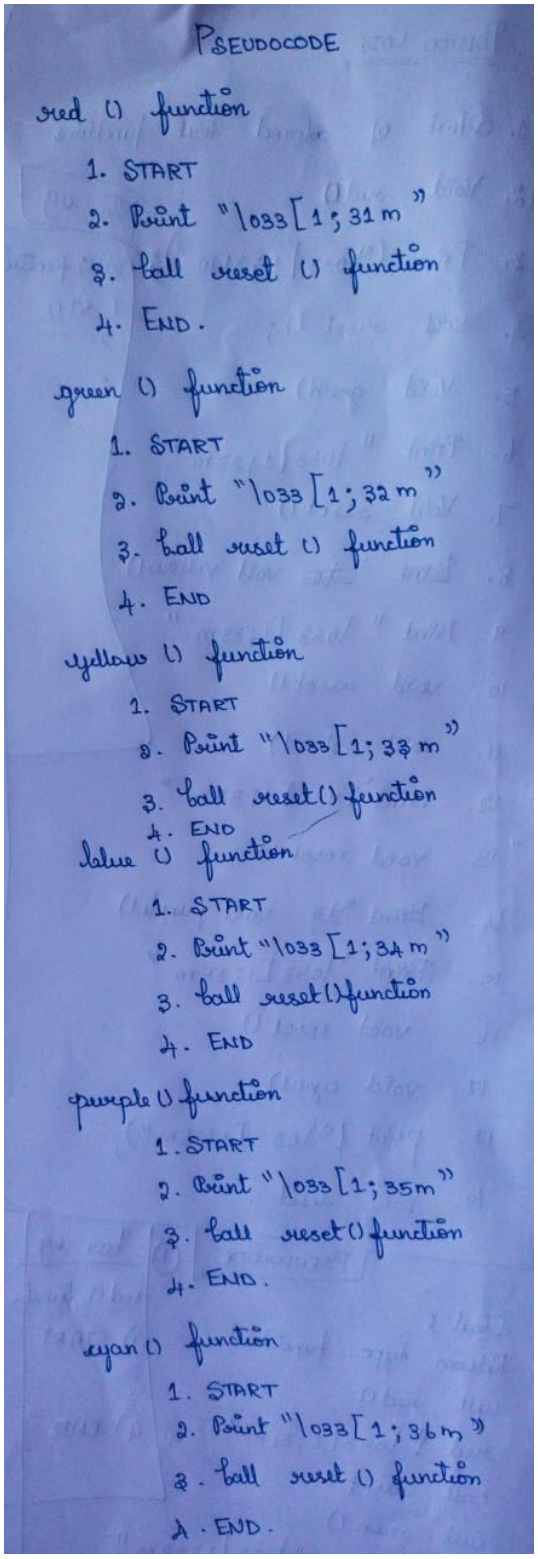
17. Print ("\n******* RULES ********\n")
Print ("******* Please Read ~~Carefully~~ ******\n")
Print ("\n1. Tokens move only along the diagonal Squares in forward direction") (\n2. Tokens move only one Square during each turn.") (\n3. Once a player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.") (\n4. You can Continue the game until you Surrender.") (\n5. Player 1's token is 'X' player 2's token is 'O'. \n6. Each player has 12 tokens on a 8x8 square board") (\n7. Row number with Prefix r and column number with prefix c will be displayed on all 4 corners of the board \n Enter Coordinates accordingly when prompted \n 8. Enter 99 as token row number and column number to Surrender \n 9. Enter 88 as token row number and column number



for help menu. \n 10. If player makes invalid move once, 2nd chance will be given. \n If player again makes invalid move, turn will go to other player. \n)
(\n ******* Press ENTER key to Continue *******)
in green colour using green() function.

48. draw_board() function
49. turn = get_turn (turn, player_X, player_O)
50. display_prev_score(player_X, player_O)
51. STOP.

Functions for colored text



PSEUDOCODE

red () function

1. START
2. Print "\033[1;31m"
3. call reset () function
4. END.

green () function

1. START
2. Print "\033[1;32m"
3. call reset () function
4. END

yellow () function

1. START
2. Print "\033[1;33m"
3. call reset() function
4. END

blue () function

1. START
2. Print "\033[1;34m"
3. call reset() function
4. END

purple () function

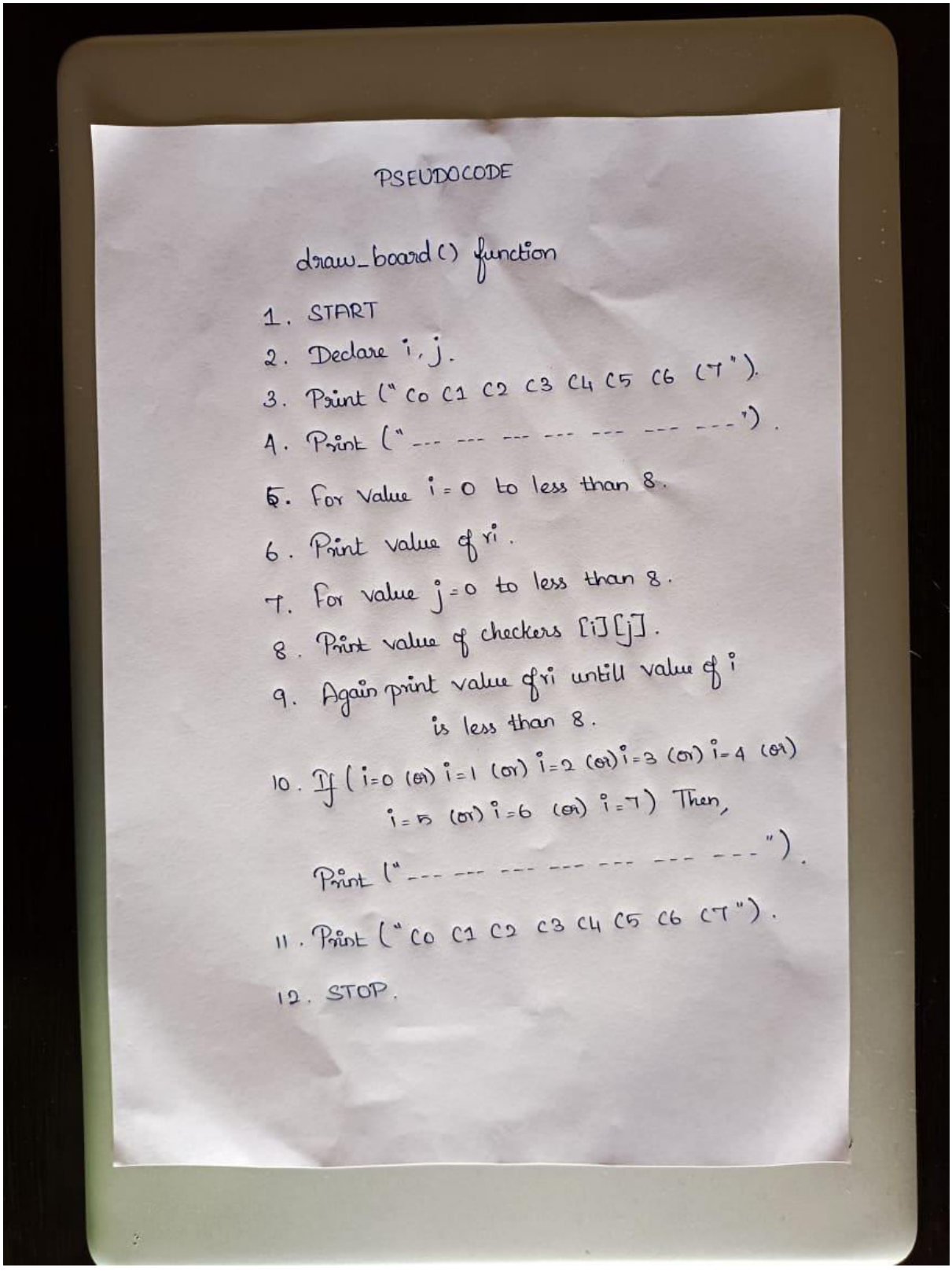
1. START
2. Print "\033[1;35m"
3. call reset() function
4. END.

cyan () function

1. START
2. Print "\033[1;36m"
3. call reset () function
4. END.

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Function for drawing checker board and displaying tokens



PSEUDOCODE

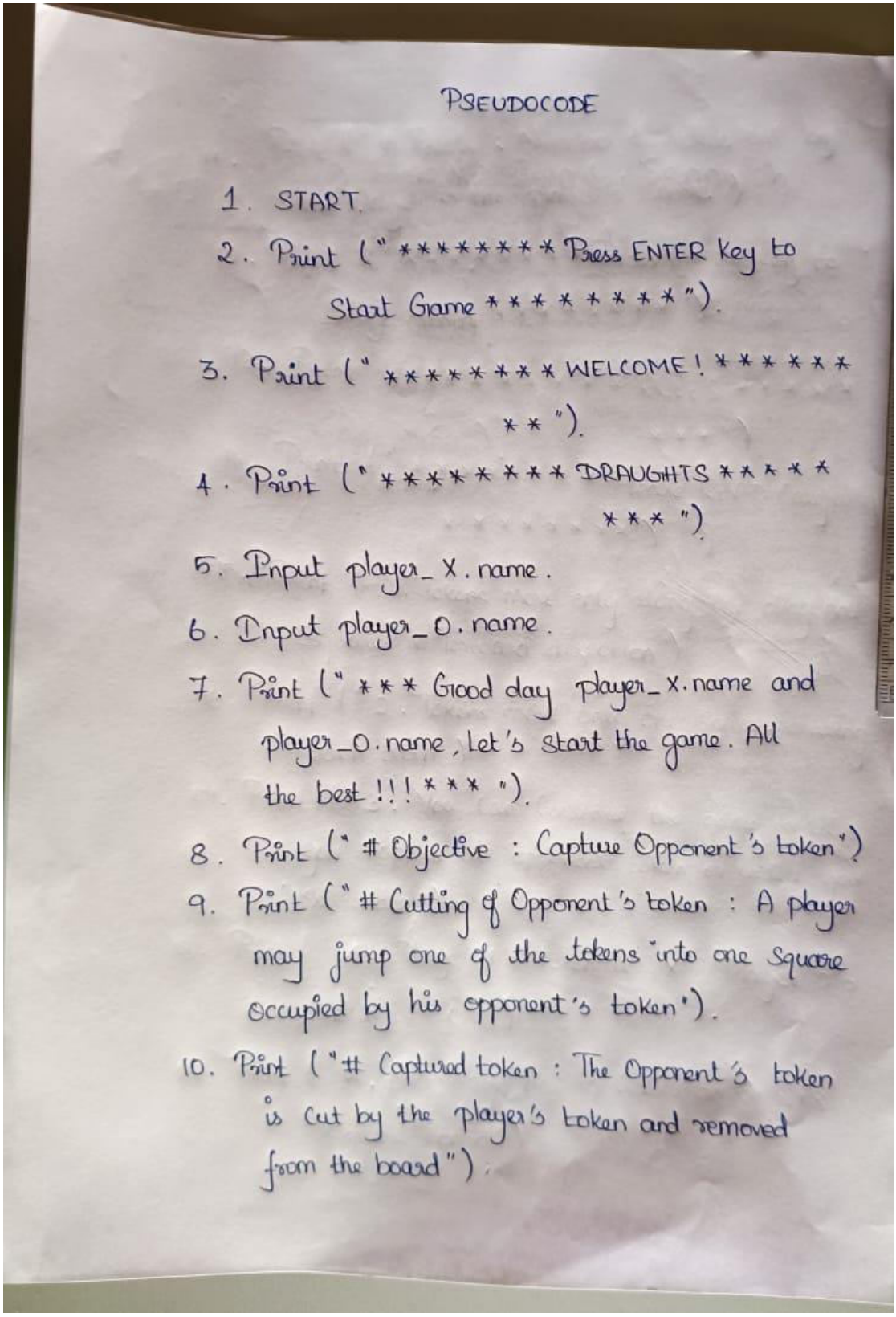
draw_board() function

1. START
2. Declare i, j.
3. Print (" C0 C1 C2 C3 C4 C5 C6 C7 ").
4. Print (" --- --- --- --- --- --- --- ").
5. For value i = 0 to less than 8.
6. Print value of ri.
7. For value j = 0 to less than 8.
8. Print value of checkers [i][j].
9. Again print value of ri untill value of i is less than 8.
10. If (i=0 (or) i=1 (or) i=2 (or) i=3 (or) i=4 (or) i=5 (or) i=6 (or) i=7) Then,

    Print (" --- --- --- --- --- --- --- ").
11. Print (" C0 C1 C2 C3 C4 C5 C6 C7 ").
12. STOP.

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

For Welcome screen, player input, and display of objectives and rules in main() function-



PSEUDOCODE

1. START.
2. Print ("******** Press ENTER key to Start Game ********").
3. Print ("******** WELCOME! ******** **").
4. Print ("********* DRAUGHTS ***** ***").
5. Input player_X.name.
6. Input player_O.name.
7. Print ("*** Good day player_X.name and player_O.name, let's start the game. All the best !!! *** ").
8. Print ("# Objective : Capture Opponent's token")
9. Print ("# Cutting of Opponent's token : A player may jump one of the tokens into one square occupied by his opponent's token").
10. Print ("# Captured token : The Opponent's token is cut by the player's token and removed from the board").

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

11. Print ("# Points : Number of token cut by the player is the player's score").

12. Print ("******** Press ENTER Key to Continue ********").

13. Print ("******** RULES ***** ***").

14. Print ("******** Please READ Carefully ********").

15. Print ("1. Tokens move only along the diagonal Squares in forward direction").

16. Print ("2. Tokens move only one Square during each turn").

17. Print ("3. You can Continue the game until you capture all opponent token or untill you Surrender").

18. Print ("4. Player 1's token is 'X'; Player 2's token is 'O'").

19. Print ("5. Each player has 12 tokens on a 8x8 Square board")

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

20. Print ("6. Row number with prefix r and Column number with prefix c will be displayed on all 4 corners of the board").

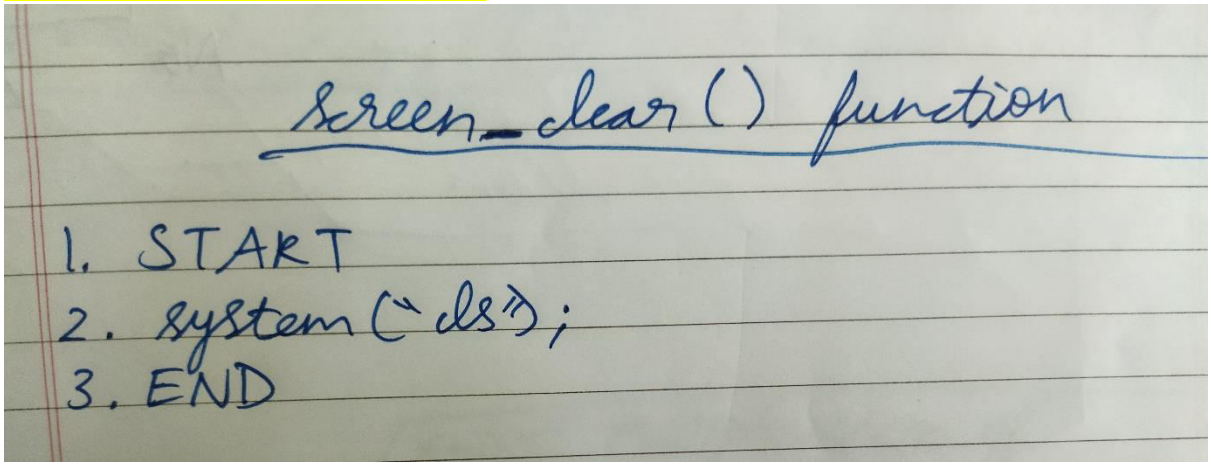
21. Print ("7. Enter 9 9 as token coordinates to Surrender").

22. Print ("8. Enter 8 8 as token coordinates for help menu").

23. Print ("******** Press ENTER key to Continue ********").

24. STOP.

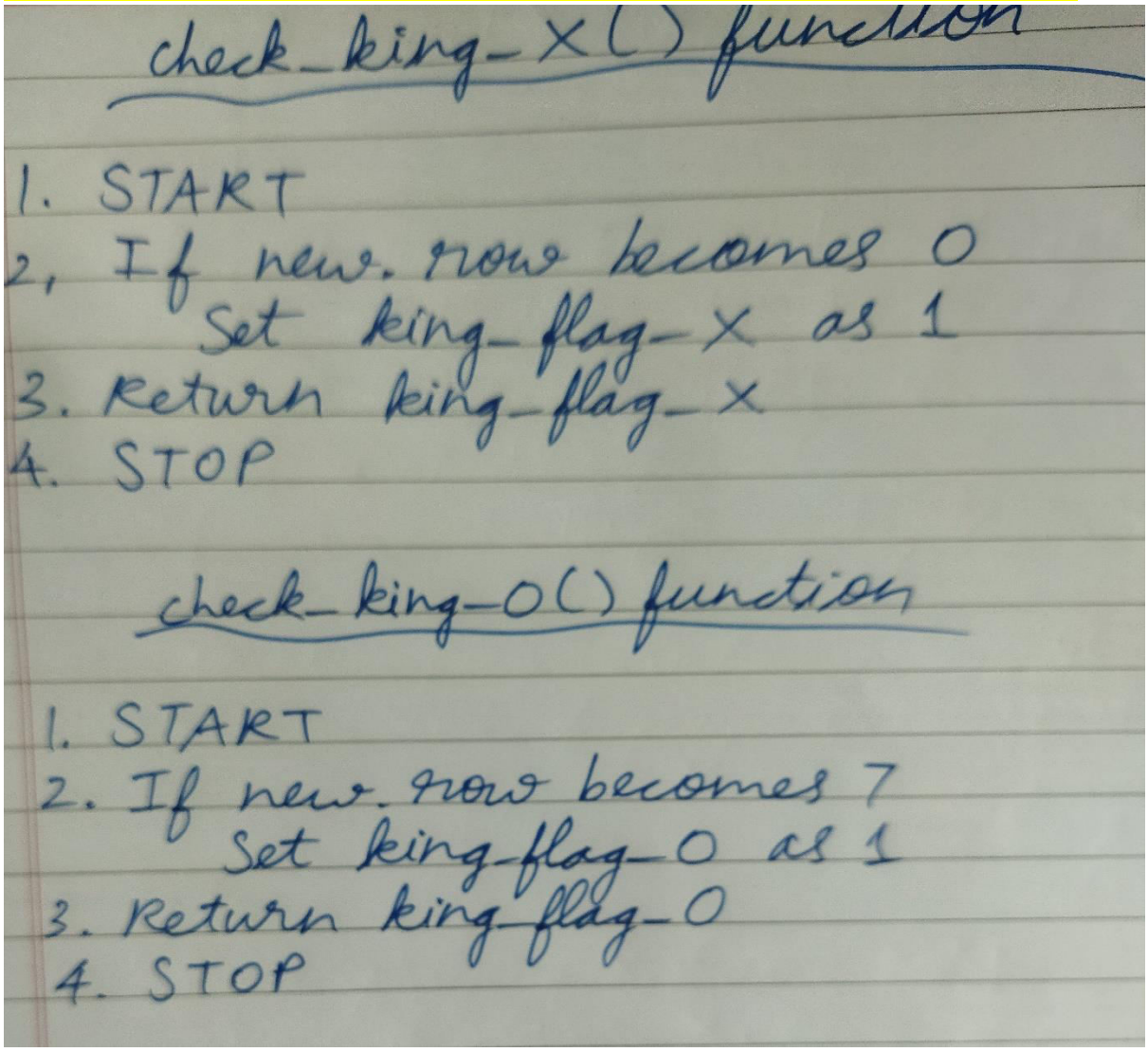
Function for clearing screen



screen_clear() function

1. START
2. system("cls");
3. END

Functions to check for kings for player 1 and player 2 respectively



check_king_X() function

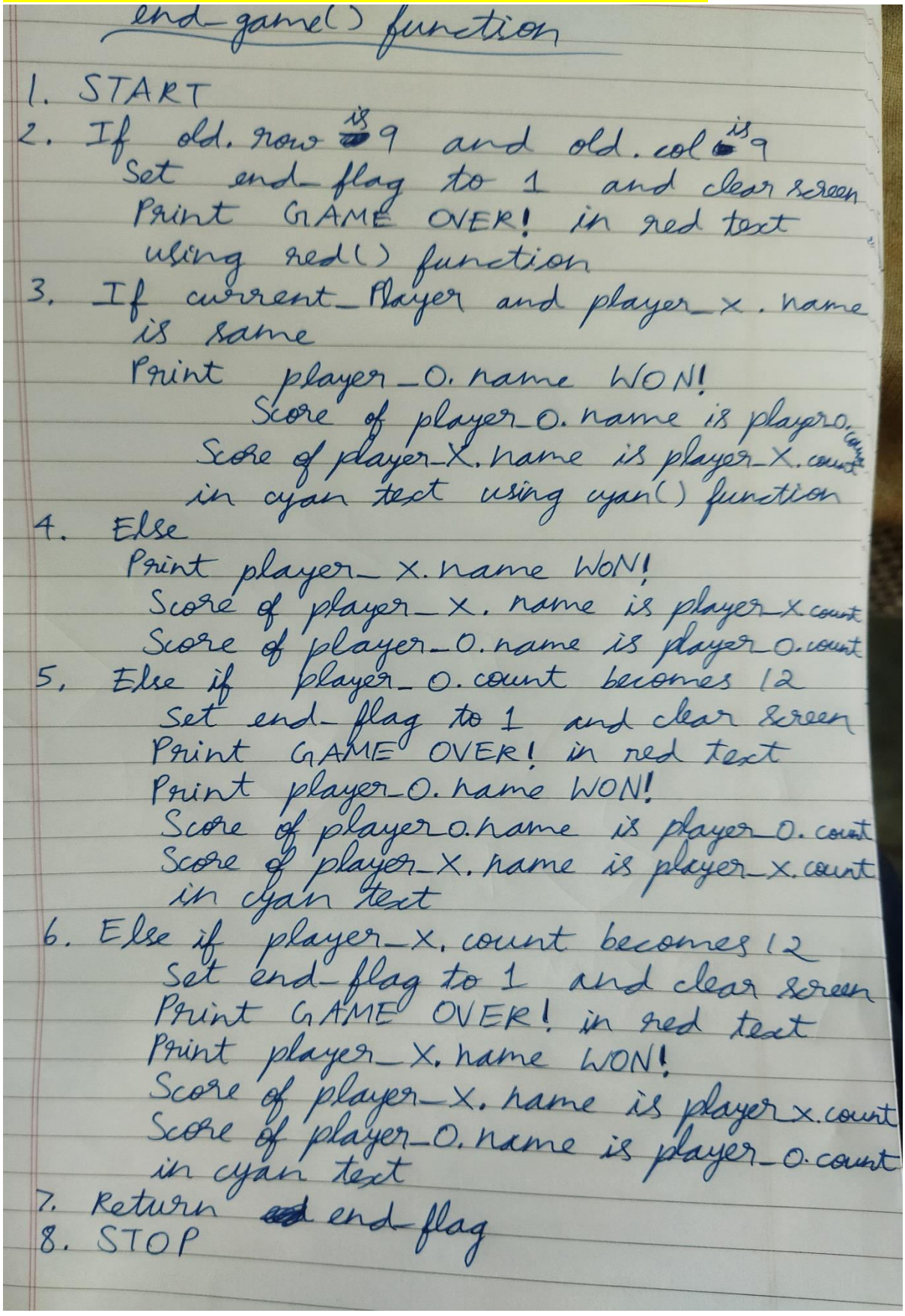
1. START
2. If new.row becomes 0
   Set king_flag_X as 1
3. Return king_flag_X
4. STOP

check_king_O() function

1. START
2. If new.row becomes 7
   Set king_flag_O as 1
3. Return king_flag_O
4. STOP

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Function for ending game and displaying player scores



end_game() function

1. START
2. If old.row is 9 and old.col is 9
   Set end_flag to 1 and clear screen
   Print GAME OVER! in red text
   using red() function
3. If current_Player and player_x.name
   is same
   Print player_O.name WON!
   Score of player_O.name is player_O.
   Score of player_X.name is player_X.count
   in cyan text using cyan() function
4. Else
   Print player_X.name WON!
   Score of player_X.name is player_X.count
   Score of player_O.name is player_O.count
5. Else if player_O.count becomes 12
   Set end_flag to 1 and clear screen
   Print GAME OVER! in red text
   Print player_O.name WON!
   Score of player_O.name is player_O.count
   Score of player_X.name is player_X.count
   in cyan text
6. Else if player_X.count becomes 12
   Set end_flag to 1 and clear screen
   Print GAME OVER! in red text
   Print player_X.name WON!
   Score of player_X.name is player_X.count
   Score of player_O.name is player_O.count
   in cyan text
7. Return end_flag
8. STOP

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

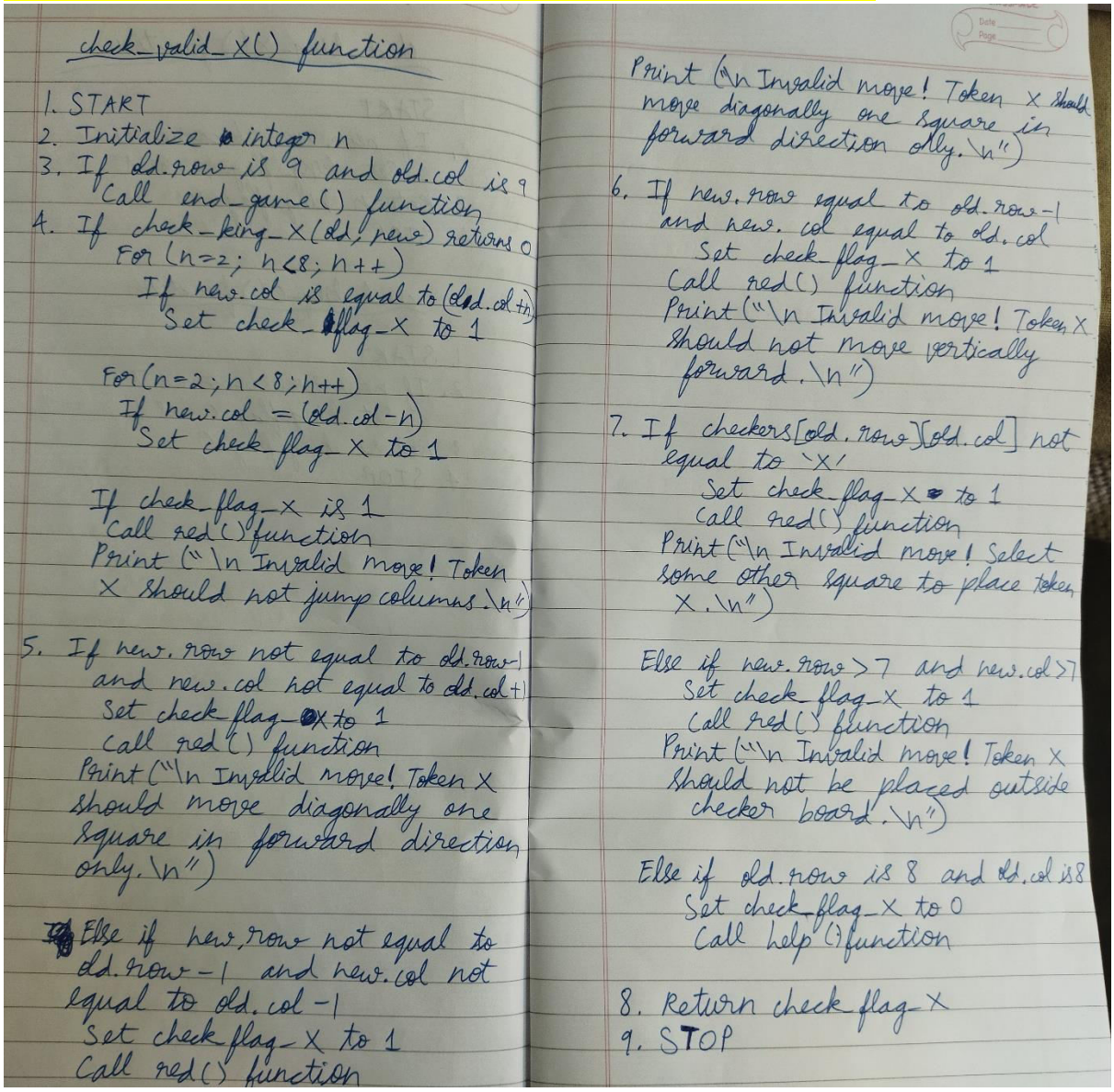
Function for calling pop up help menu

help() function

1. START
2. If old.row is 8 and old.col is 8
3. Print ("********RULES********\n");
4. Print ("********Please READ carefully **********\n")
5. Print ("\n1. Tokens move only along the diagonal squares in forward direction")
6. Print ("\n2. Tokens move only one square during each turn.")
7. Print ("\n3. Once a player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.")
8. Print ("\n He can move any of his tokens in all directions and any number of squares during bonus turn.\n But if he gets king in 2nd chance turn, bonus turn won't be given.)
9. Print ("\n4. You can continue the game until you capture all opponent tokens or until someone surrenders.")
10. Print ("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.")
11. Print ("\n7. Row number with prefix r and column number with prefix c will be displayed on all 4 corners of the board.\n Enter coordinates accordingly when prompted.")
12. Print ("\n8. If player makes invalid move once, 2nd chance will be given.\n If the player again makes invalid move, turn will go to other player.\n")
13. STOP

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Function for checking if move made by player 1 is valid



check_valid_X() function

1. START
2. Initialize integer n
3. If old.row is 9 and old.col is 9
   Call end_game() function
4. If check_king_X(old, new) returns 0
   For (n=2; n<8; n++)
   If new.col is equal to (old.col+n)
   Set check_flag_X to 1

   For(n=2; n<8; n++)
   If new.col = (old.col-n)
   Set check_flag_X to 1

   If check_flag_X is 1
   Call red() function
   Print ("\n Invalid move! Token X should not jump columns.\n")

5. If new.row not equal to old.row-1 and new.col not equal to old.col+1
   Set check_flag_X to 1
   Call red() function
   Print ("\n Invalid move! Token X should move diagonally one square in forward direction only.\n")

   Else if new.row not equal to old.row-1 and new.col not equal to old.col-1
   Set check_flag_X to 1
   Call red() function
   Print ("\n Invalid move! Token X should move diagonally one square in forward direction only.\n")

6. If new.row equal to old.row-1 and new.col equal to old.col
   Set check_flag_X to 1
   Call red() function
   Print ("\n Invalid move! Token X should not move vertically forward.\n")

7. If checkers[old.row][old.col] not equal to 'X'
   Set check_flag_X to 1
   Call red() function
   Print ("\n Invalid move! Select some other square to place token X.\n")

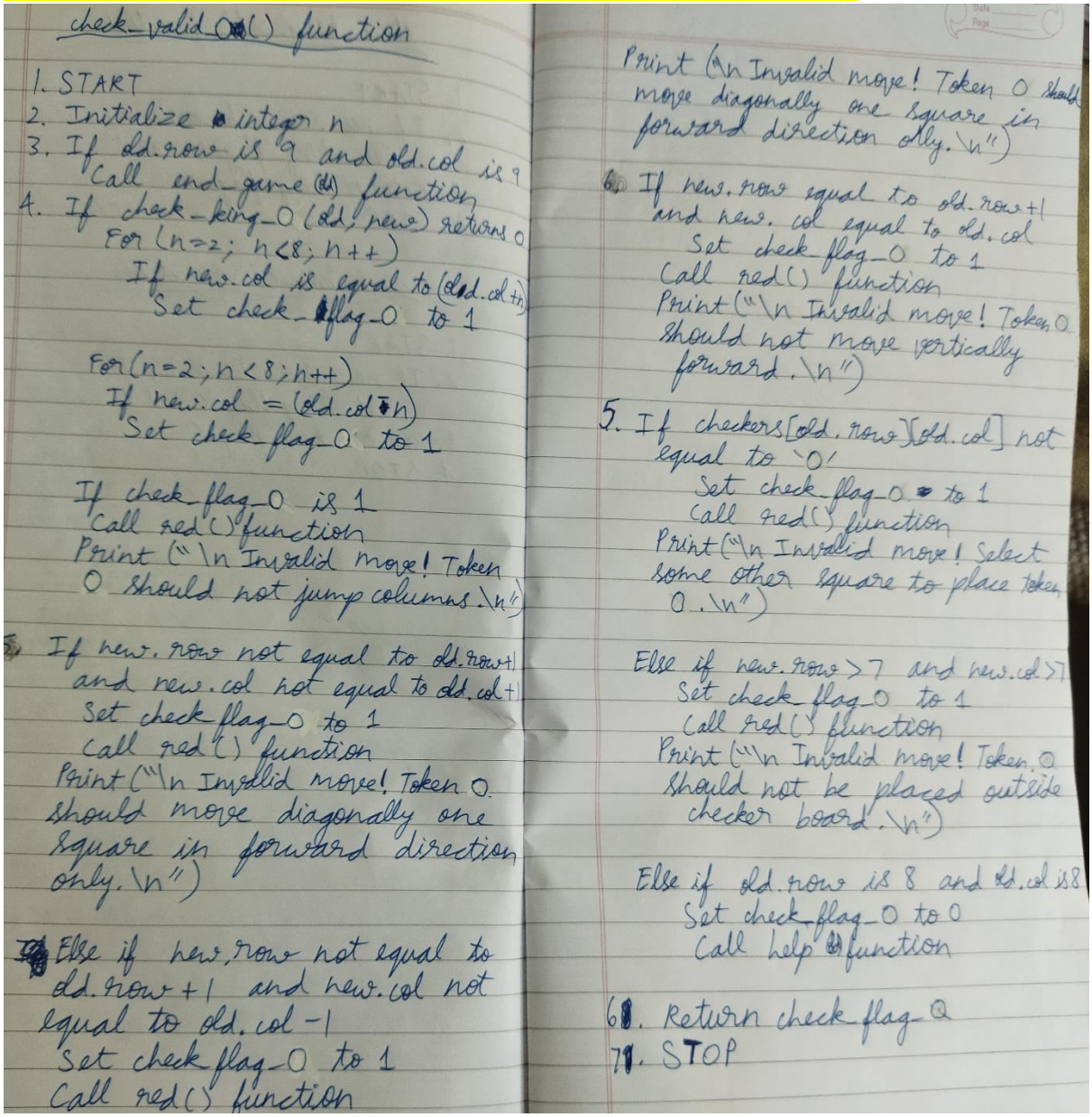
   Else if new.row > 7 and new.col > 7
   Set check_flag_X to 1
   Call red() function
   Print ("\n Invalid move! Token X should not be placed outside checker board.\n")

   Else if old.row is 8 and old.col is 8
   Set check_flag_X to 0
   Call help() function

8. Return check_flag_X
9. STOP

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Function for checking if move made by player 2 is valid



check_valid() function

1. START
2. Initialize integer n
3. If old.row is 9 and old.col is 9
   Call end_game() function
4. If check_king_O (old, new) returns 0
   For (n=2; n<8; n++)
   If new.col is equal to (old.col+n)
   Set check_flag_O to 1

   For (n=2; n<8; n++)
   If new.col = (old.col-n)
   Set check_flag_O to 1

   If check_flag_O is 1
   Call red() function
   Print ("\n Invalid move! Token O should not jump columns.\n")

5. If new.row not equal to old.row+1 and new.col not equal to old.col+1
   Set check_flag_O to 1
   Call red() function
   Print ("\n Invalid move! Token O should move diagonally one square in forward direction only.\n")

   Else if new.row not equal to old.row+1 and new.col not equal to old.col-1
   Set check_flag_O to 1
   Call red() function
   Print (\n Invalid move! Token O should move diagonally one square in forward direction only.\n")

6. If new.row equal to old.row+1 and new.col equal to old.col
   Set check_flag_O to 1
   Call red() function
   Print ("\n Invalid move! Token O should not move vertically forward.\n")

5. If checkers[old.row][old.col] not equal to 'O'
   Set check_flag_O to 1
   Call red() function
   Print ("\n Invalid move! Select some other square to place token O.\n")

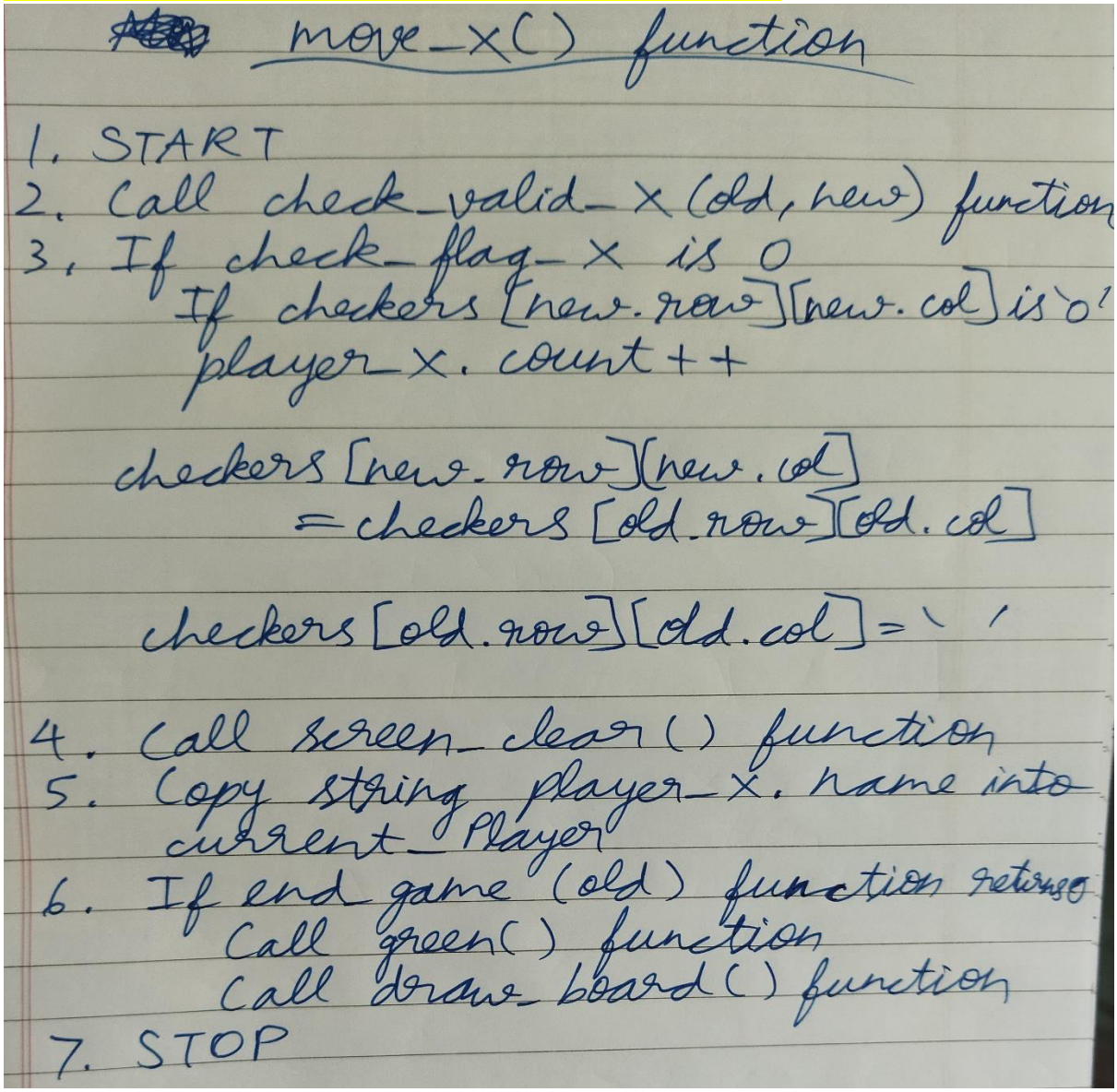
   Else if new.row > 7 and new.col > 7
   Set check_flag_O to 1
   Call red() function
   Print ("\n Invalid move! Token O should not be placed outside checker board.\n")

   Else if old.row is 8 and old.col is 8
   Set check_flag_O to 0
   Call help() function

6. Return check_flag_O
7. STOP

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Function to move X tokens on the checker board



move_x() function

1. START
2. Call check_valid_x (old, new) function
3. If check_flag_x is 0
   If checkers [new.row][new.col] is '0'
   player_x.count++

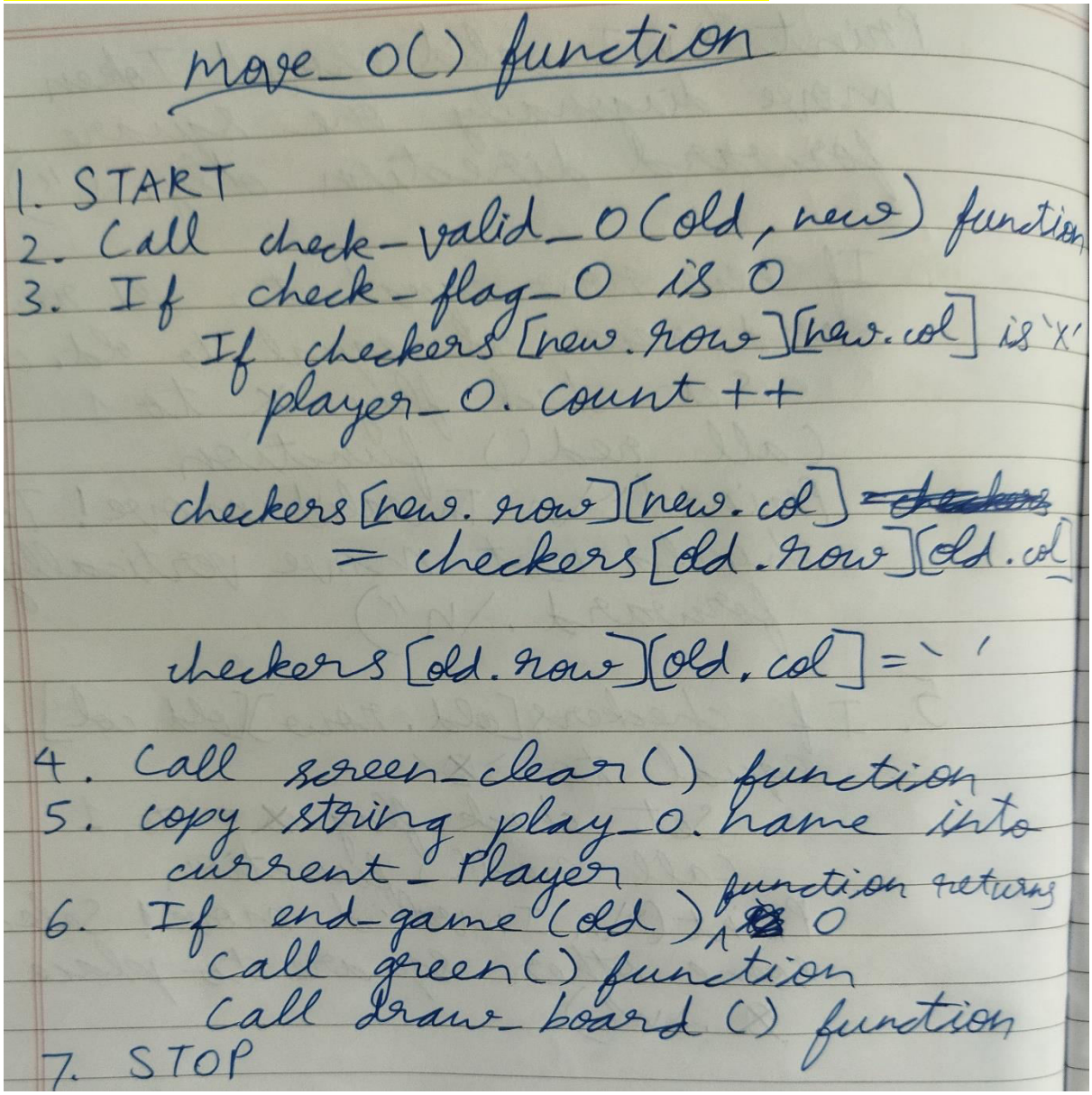
   checkers [new.row][new.col]
   = checkers [old.row][old.col]

   checkers [old.row][old.col] = ' '

4. Call screen_clear() function
5. Copy string player_x.name into current_Player
6. If end_game (old) function returns 0
   Call green() function
   Call draw_board() function
7. STOP

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Function to move O tokens on the checker board



move_O() function

1. START
2. Call check-valid_O(old, new) function
3. If check-flag_O is 0
   If checkers[new.row][new.col] is 'X'
   player_O.count ++

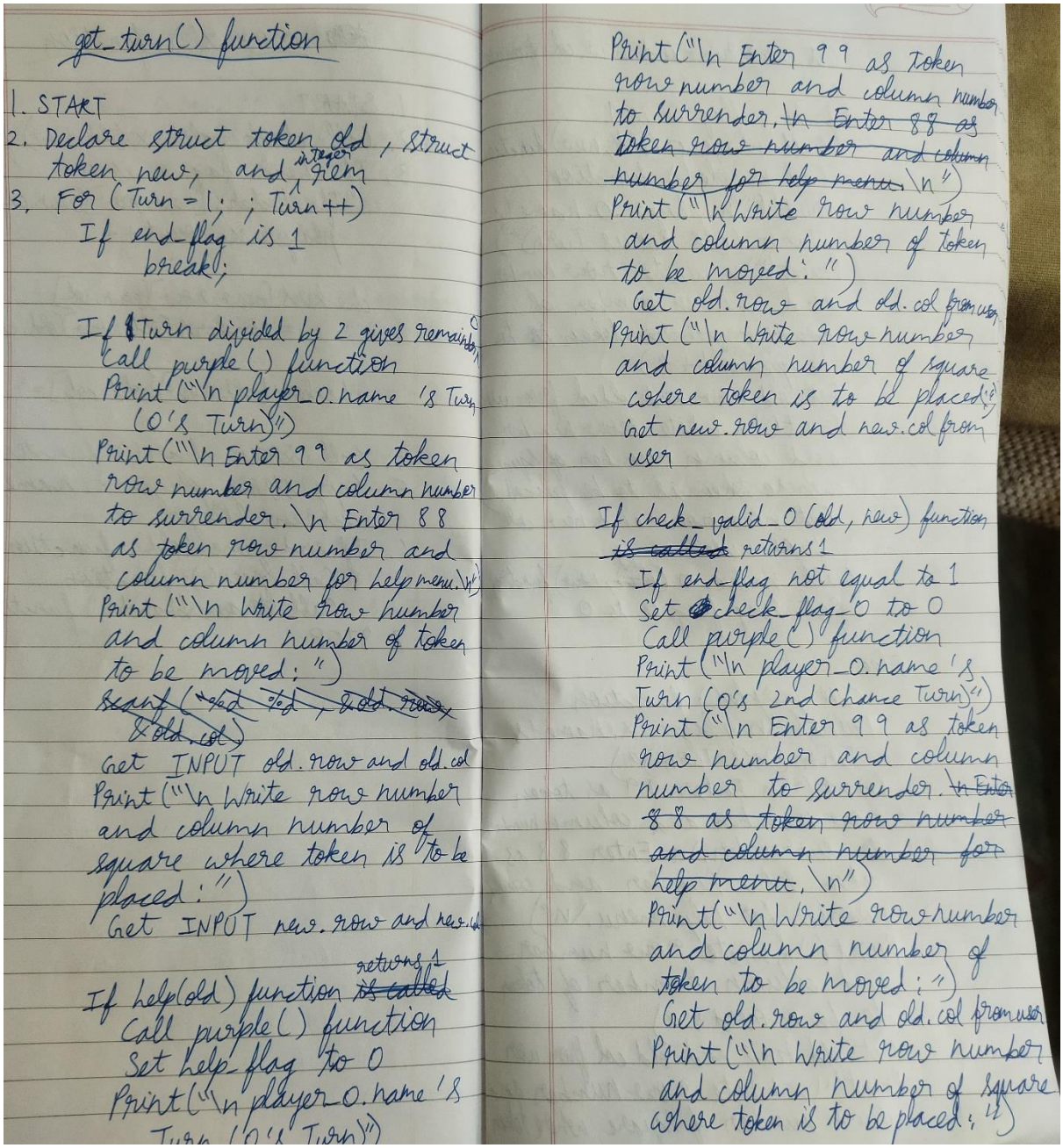
   checkers[new.row][new.col] ~~= checkers~~
   = checkers[old.row][old.col]

   checkers[old.row][old.col] = ' '

4. Call screen_clear() function
5. copy string play_O.name into current_player
6. If end_game(old) function returns 0
   call green() function
   call draw_board() function
7. STOP

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Function to alternate turns between the 2 players and keep the game running



get_turn() function

1. START
2. Declare struct token old, struct token new, and integer item
3. For (Turn = 1; ; Turn++)
   If end_flag is 1
   break;

   If Turn divided by 2 gives remainder 1
   Call purple() function
   Print("\n player_0.name 's Turn (O's Turn)")
   Print("\n Enter 99 as token row number and column number to surrender. \n Enter 88 as token row number and column number for help menu.\n")
   Print("\n Write row number and column number of token to be moved: ")
   ~~Scanf("%d %d", &old.row, &old.col)~~
   Get INPUT old.row and old.col
   Print("\n Write row number and column number of square where token is to be placed: ")
   Get INPUT new.row and new.col

   If help(old) function ~~is called~~ returns 1
   Call purple() function
   Set help_flag to 0
   Print("\n player_0.name 's Turn (O's Turn)")

   Print("\n Enter 99 as token row number and column number to surrender, \n Enter 88 as ~~token row number and column number for help menu~~ \n")
   Print("\n Write row number and column number of token to be moved: ")
   Get old.row and old.col from user
   Print("\n Write row number and column number of square where token is to be placed: ")
   Get new.row and new.col from user

   If check_valid_0 (old, new) function ~~is called~~ returns 1
   If end_flag not equal to 1
   Set check_flag_0 to 0
   Call purple() function
   Print("\n player_0.name 's Turn (O's 2nd Chance Turn)")
   Print("\n Enter 99 as token row number and column number to surrender. ~~\n Enter 88 as token row number and column number for help menu~~. \n")
   Print("\n Write row number and column number of token to be moved: ")
   Get old.row and old.col from user
   Print("\n Write row number and column number of square where token is to be placed: ")

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

Get new_row and new_col from user

If king_flag_O is 1
Call move_O(old, new) function
Call purple() function
Print("\n player_O.name's Turn (O's Bonus Turn)")
Print("\n Write row number and column number of ~~square where~~ token ~~to~~ to be ~~placed~~ moved:")
Get old_row and old_col from user
Print("\n Write row number and column number of square where token is to be placed:")
Get new_row and new_col from user
Call move_O(old, new) function
Set king_flag_O to 0

Else
Call purple() function
Print("\n player_X.name's Turn (X's Turn)")
Print("\n Enter 99 as token row number and column number to surrender. \n Enter 88 as token row number and column number for help menu.\n")
Print("\n Write row number and column number of token to be moved:")
Get old_row and old_col from user
Print("\n Write row number and column number of square where token is to be placed:")
Get new_row and new_col from user

If help(old) function returns 1
Set help_flag to 0
call purple() function
Print("\n player_X.name's Turn (X's Turn)")
Print("\n Enter 99 as token row number and column number to surrender.")
Print("\n Write row number and column number of token to be moved:")
Get old_row and old_col from user
Print("\n Write row number and column number of square where token is to be placed:")
Get new_row and new_col from user

If check_valid_X(old, new) returns 1
If end_flag not equal to 1
Set check_flag_X to 0
call purple() function
Print("\n player_X.name's Turn (X's 2nd Chance Turn)")
Print("\n Enter 99 as token row number and column number to surrender.")
Print("\n Write row number and column number of token to be moved:")

Get old.row and old.col from user
Print("\n Write row number and column number of square where token is to be placed:")
Get new.row and new.col from user

If king_flag_x is 1
Call move_x(old, new) function
Print("\n player_x.name's Turn (x's Bonus Turn)")
Print("\n Write row number and column number of token to be moved:")
Get old.row and old.col from user
Print("\n Write row number and column number of square where token is to be placed:")
Get new.row and new.col from user

Call move_x(old, new) function
Set king_flag_x to 0
Set rem to 1

4. Return rem
5. STOP

For storing and displaying scores of previous players using files

1. START
2. Opening a file in append mode and store in *fptr
3. Check fptr is null or not.
4. If it is null, print error and exit from program abnormally
5. If it is not null, append the player_O, player_X. name and count using fprintf in the text file (scoreboard.txt)
6. Close the file fptr by fclose
7. To sort the current players scores with previous leaderboard scores.
8. Define a player structure with char name, int name, variables.
9. Initialize i=0, j=0, nli=0
10. char file name[100] = d://scoreboard.txt, char c=0.
11. Now, open the file in read mode with *fp as pointer
12. Check if *fp is null or not.
13. If it is null, print error and exit from the program abnormally.
14. If it is not null, check for number of lines in the file by c=getc (fp: c!=EOF: c=getc (fp))
15. If character is found in next line, nli increments 1 to count number of lines in the file.
16. Close the file using fclose (fp)
17. Then, Open file in read mode and store in * scoreboardfile1.
18. In for loop read scores from text file line by

Samuela Abigail, Sakthi Abinaya, Merlin Might, Deepiga
AI & DS Department

line using fscanf.

19. From the read value using nested for loop, sort the scores in descending order

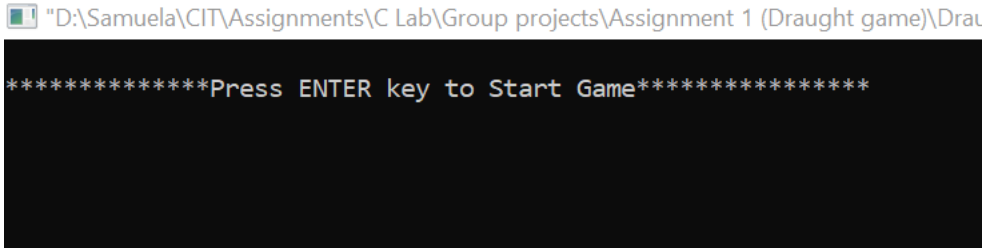
20. Close the scoreboardfile1 using fclose

21. Using for loop display the scores of the top 10 players

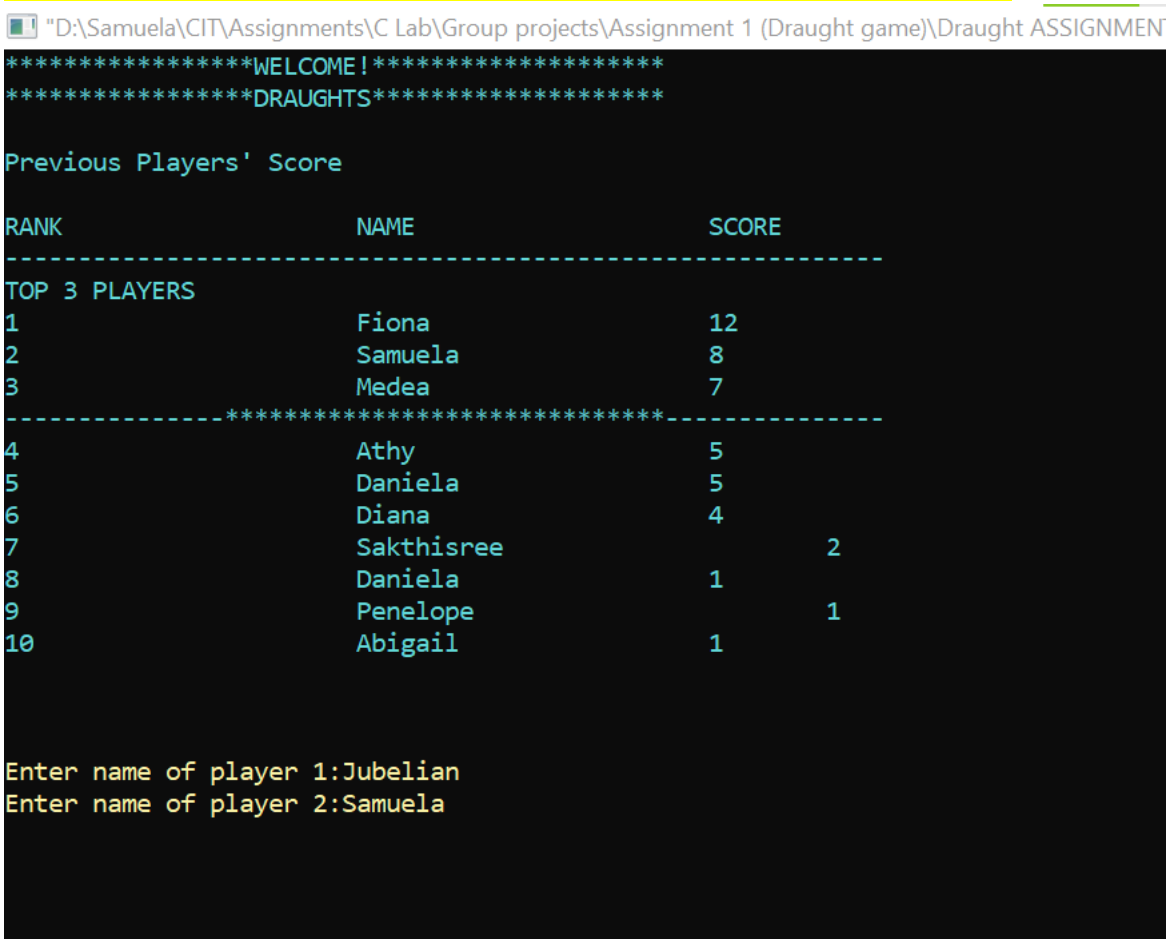
22. END.

Working screenshots:

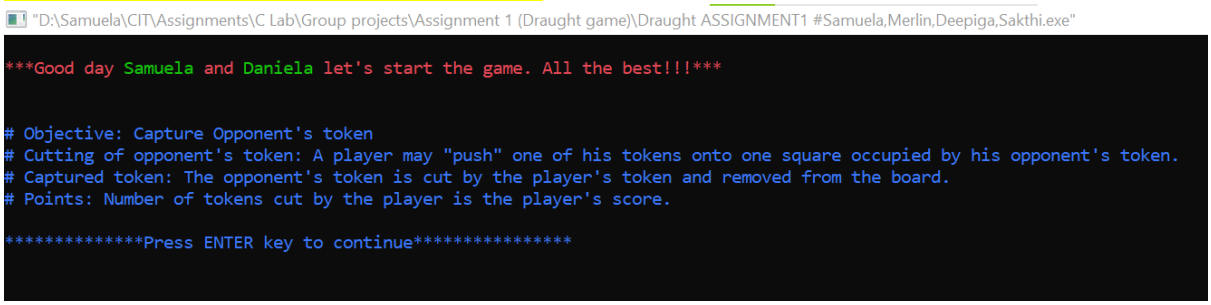
Starting of game

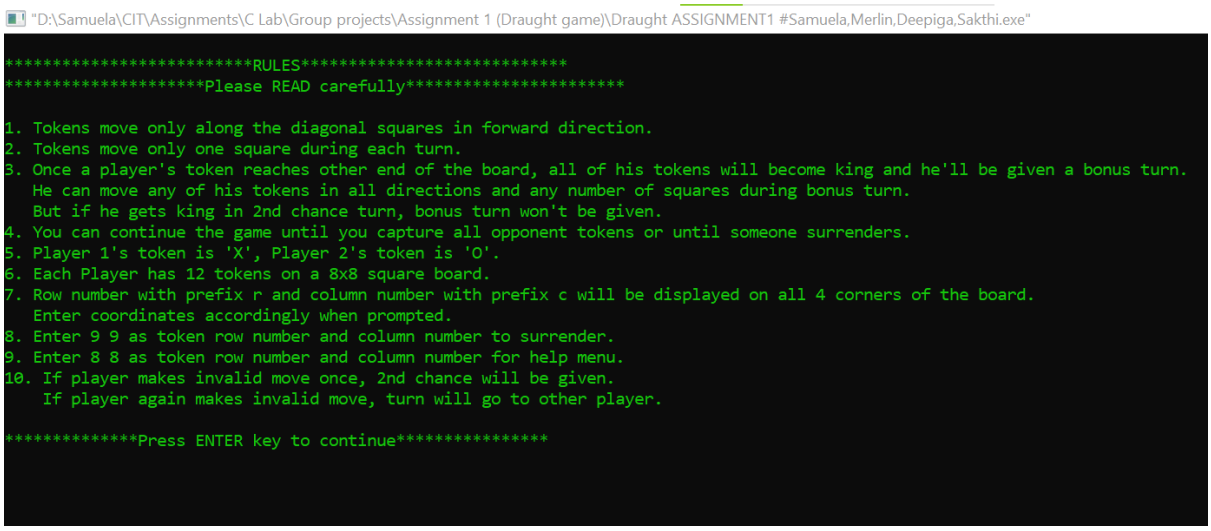


Player names input and display of previous players' scores

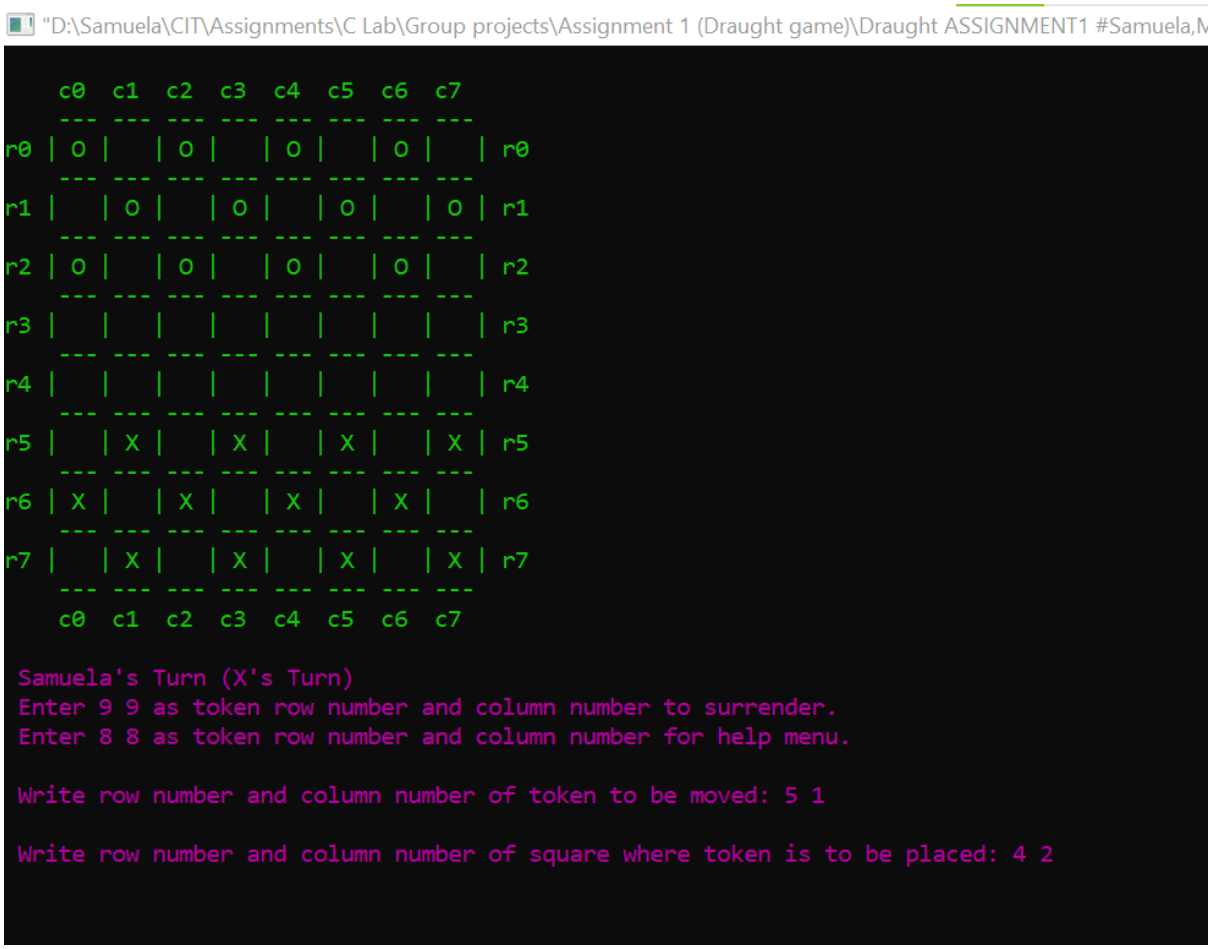


Display of Objectives and Rules

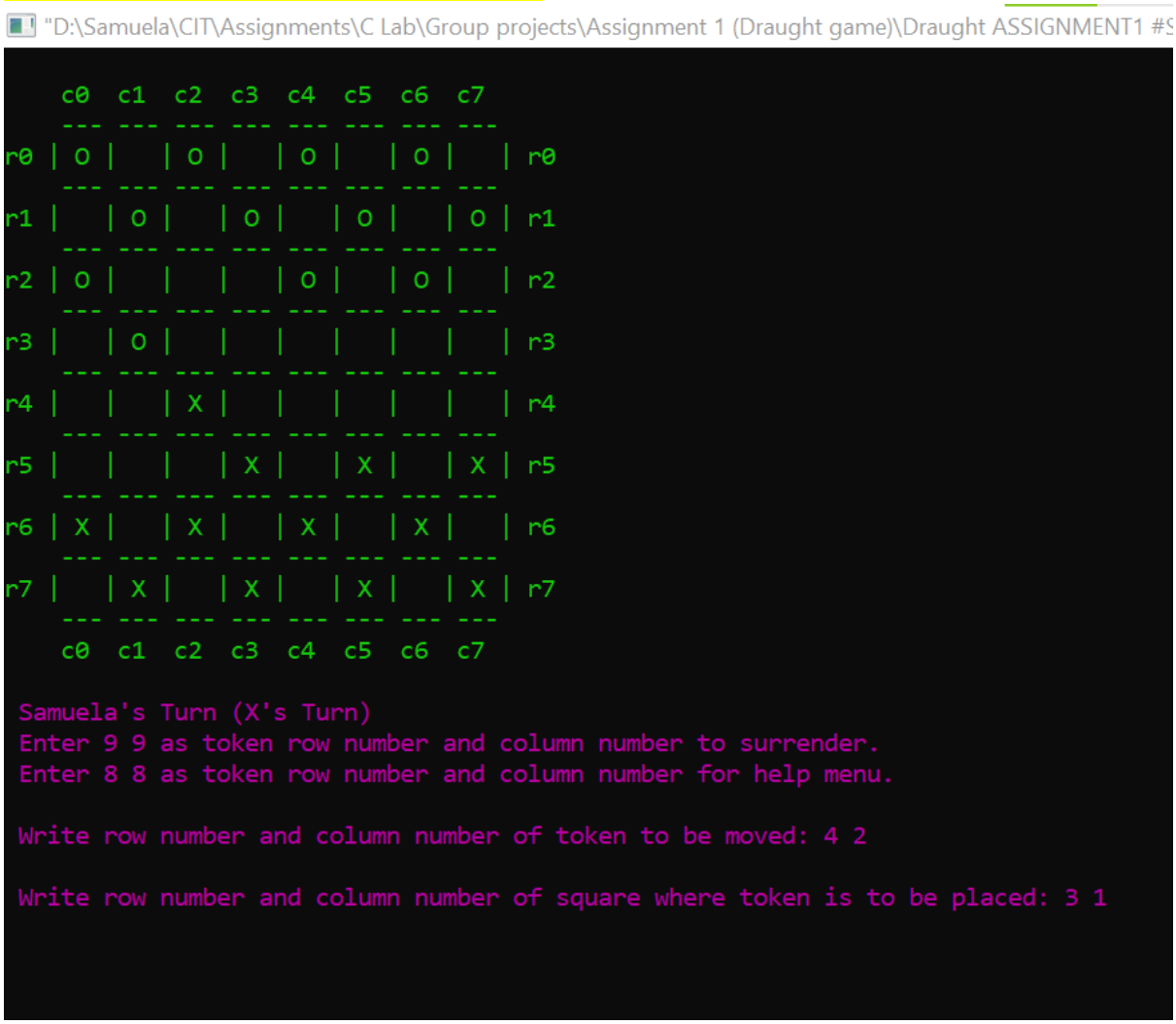




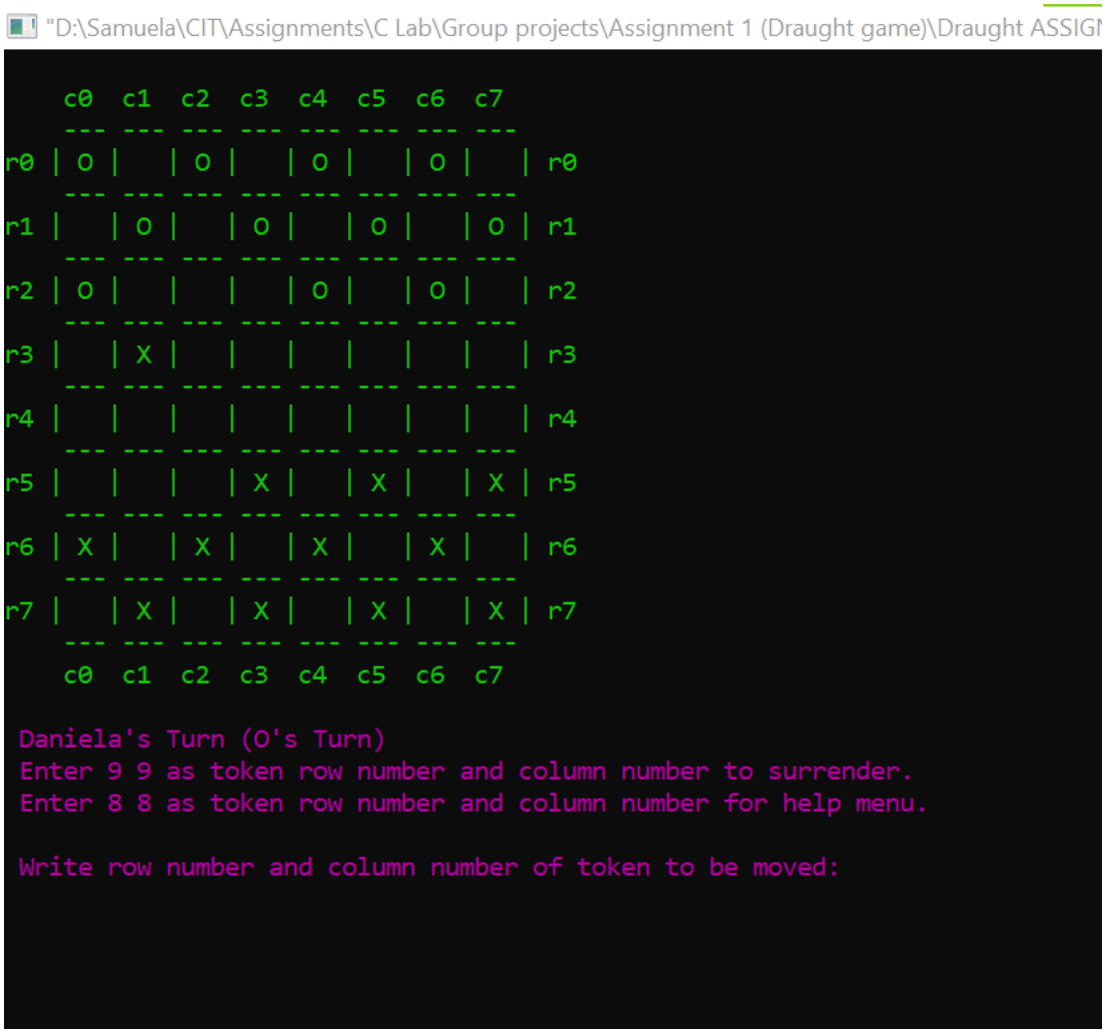
Game starts. Turn 1 belongs to Samuela (player 1) and she enters row number and column number to move X token


"D:\Samuela\CIT\Assignments\C Lab\Group projects\Assignment 1 (Draught game)\Draught ASSIGNMENT1 #Samuela,M
c0 c1 c2 c3 c4 c5 c6 c7
r0 | O | | O | | O | | O | | r0
r1 | | O | | O | | O | | O | r1
r2 | O | | O | | O | | O | | r2
r3 | | | | | | | | | r3
r4 | | | | | | | | | r4
r5 | | X | | X | | X | | X | r5
r6 | X | | X | | X | | X | | r6
r7 | | X | | X | | X | | X | r7
c0 c1 c2 c3 c4 c5 c6 c7
Samuela's Turn (X's Turn)
Enter 9 9 as token row number and column number to surrender.
Enter 8 8 as token row number and column number for help menu.
Write row number and column number of token to be moved: 5 1
Write row number and column number of square where token is to be placed: 4 2

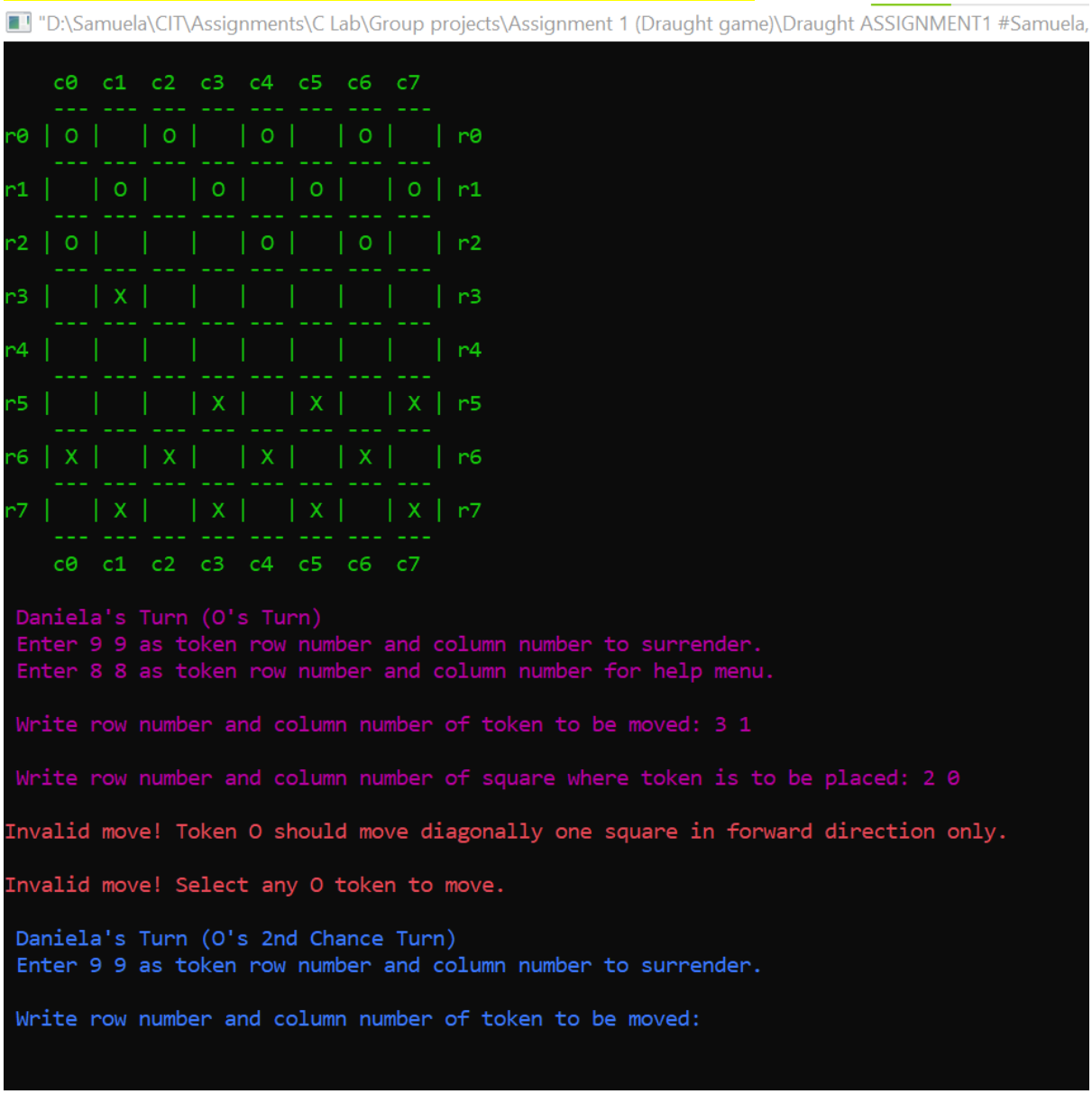
Samuela is about to cut token O



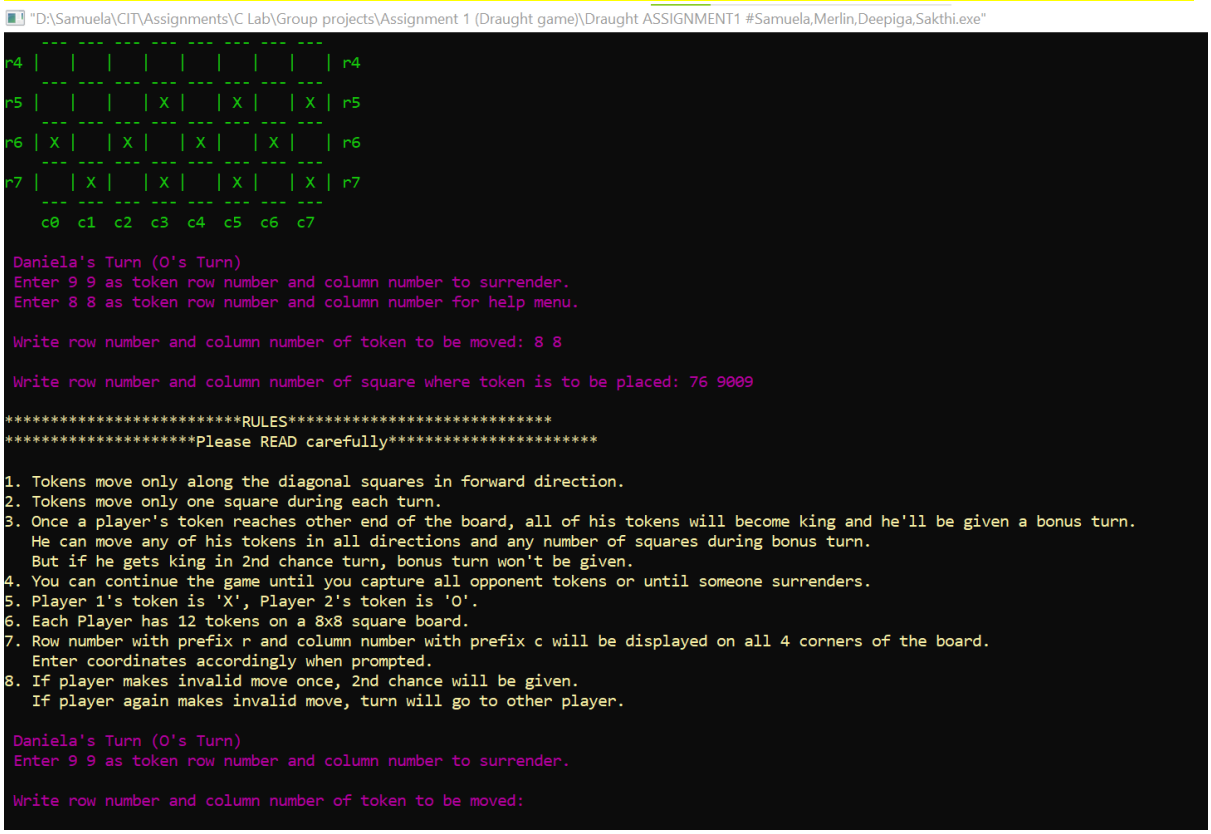
Samuela cuts O token and turn goes to Daniela. Daniela also is about to cut X token



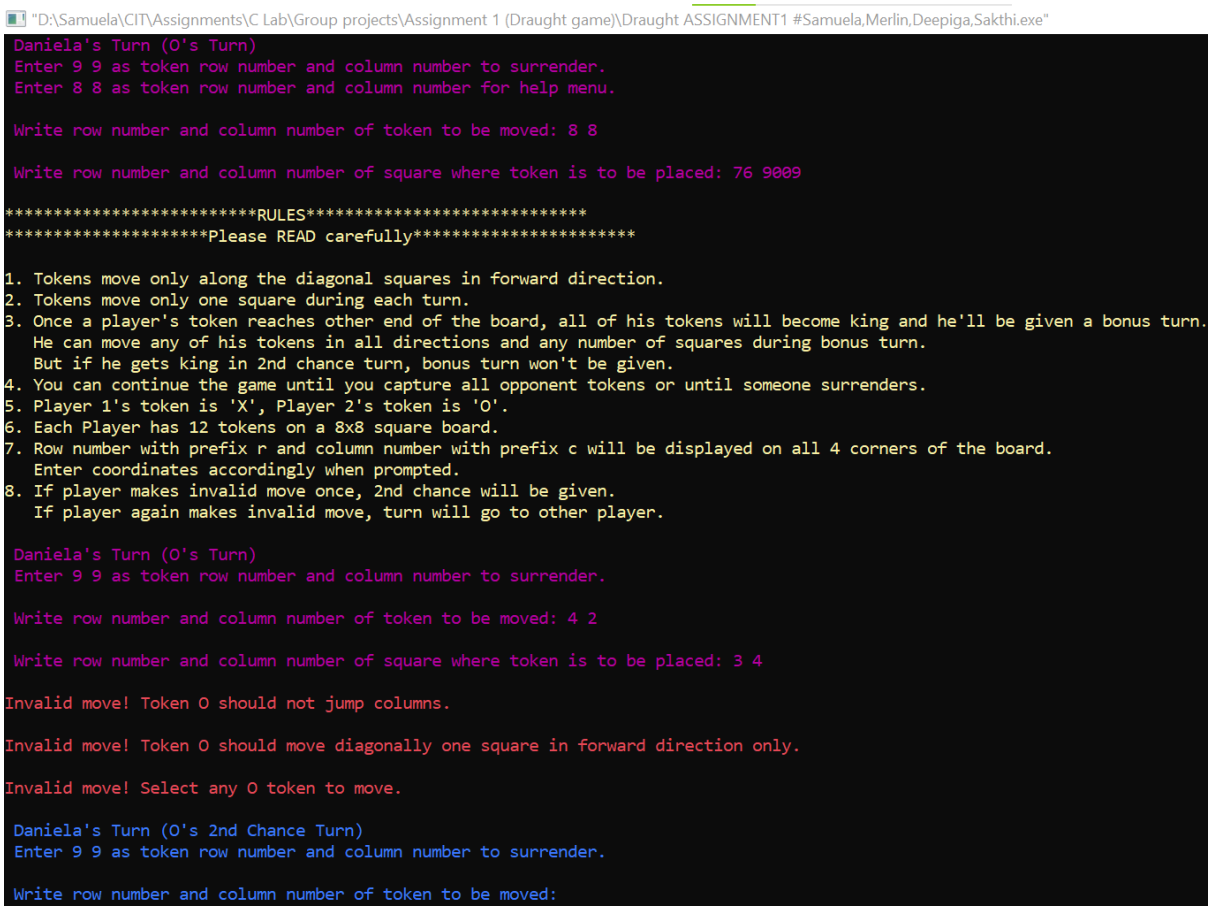
Daniela makes invalid move and gets 2nd chance



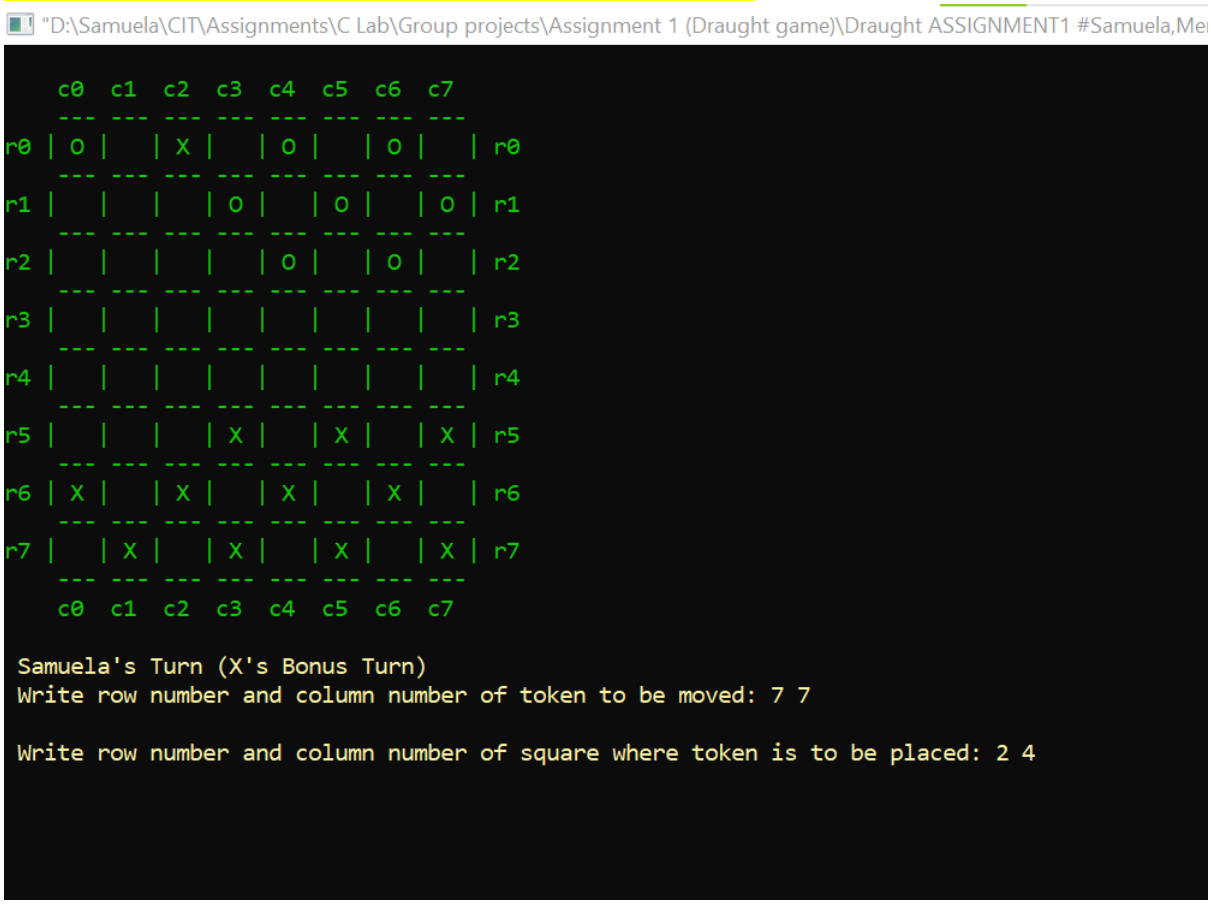
Daniela presses 8 8 to access help menu and is given another chance



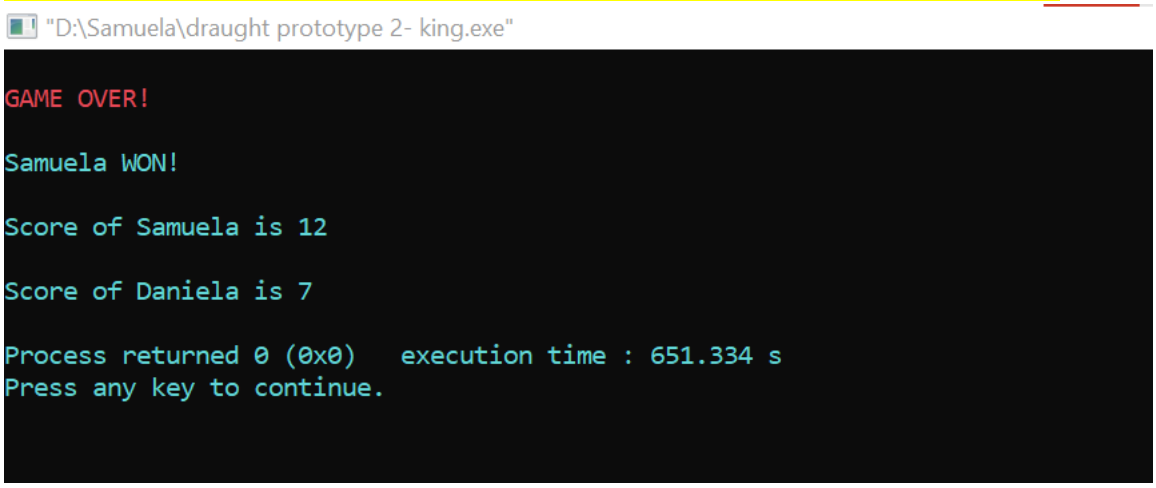
Daniela makes invalid move after accessing help menu, and is given 2nd chance


"D:\Samuela\CIT\Assignments\C Lab\Group projects\Assignment 1 (Draught game)\Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.exe"
Daniela's Turn (O's Turn)
Enter 9 9 as token row number and column number to surrender.
Enter 8 8 as token row number and column number for help menu.
Write row number and column number of token to be moved: 8 8
Write row number and column number of square where token is to be placed: 76 9009
**********RULES**********
**********Please READ carefully**********
1. Tokens move only along the diagonal squares in forward direction.
2. Tokens move only one square during each turn.
3. Once a player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.
He can move any of his tokens in all directions and any number of squares during bonus turn.
But if he gets king in 2nd chance turn, bonus turn won't be given.
4. You can continue the game until you capture all opponent tokens or until someone surrenders.
5. Player 1's token is 'X', Player 2's token is 'O'.
6. Each Player has 12 tokens on a 8x8 square board.
7. Row number with prefix r and column number with prefix c will be displayed on all 4 corners of the board.
Enter coordinates accordingly when prompted.
8. If player makes invalid move once, 2nd chance will be given.
If player again makes invalid move, turn will go to other player.
Daniela's Turn (O's Turn)
Enter 9 9 as token row number and column number to surrender.
Write row number and column number of token to be moved: 4 2
Write row number and column number of square where token is to be placed: 3 4
Invalid move! Token O should not jump columns.
Invalid move! Token O should move diagonally one square in forward direction only.
Invalid move! Select any O token to move.
Daniela's Turn (O's 2nd Chance Turn)
Enter 9 9 as token row number and column number to surrender.
Write row number and column number of token to be moved:

Samuela activates king and gets bonus turn



Samuela wins by capturing all 12 O tokens and program ends



Samuela is about to surrender/quit

"D:\Samuela\draught prototype 2- king.exe"

```
    c0  c1  c2  c3  c4  c5  c6  c7
    --- --- --- --- --- --- --- ---
r0 | O |   | O |   | O |   | O |   | r0
    --- --- --- --- --- --- --- ---
r1 |   | O |   | O |   | O |   | O | r1
    --- --- --- --- --- --- --- ---
r2 |   |   |   |   | O |   | O |   | r2
    --- --- --- --- --- --- --- ---
r3 |   | O |   |   |   |   |   |   | r3
    --- --- --- --- --- --- --- ---
r4 |   |   |   |   |   |   |   |   | r4
    --- --- --- --- --- --- --- ---
r5 |   |   |   | X |   | X |   | X | r5
    --- --- --- --- --- --- --- ---
r6 | X |   | X |   | X |   | X |   | r6
    --- --- --- --- --- --- --- ---
r7 |   | X |   | X |   | X |   | X | r7
    --- --- --- --- --- --- --- ---
    c0  c1  c2  c3  c4  c5  c6  c7

 Samuela's Turn (X's Turn)
 Enter 9 9 as token row number and column number to surrender.
 Enter 8 8 as token row number and column number for help menu.

 Write row number and column number of token to be moved: 9 9

 Write row number and column number of square where token is to be placed: 87 89
```

Game ends and Daniela wins

"D:\Samuela\draught prototype 2- king.exe"

```
GAME OVER!

Daniela WON!

Score of Daniela is 1

Score of Samuela is 1

Process returned 0 (0x0)   execution time : 110.950 s
Press any key to continue.
```