Progress #13

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to draught programed by Samuela, Sakthi, Merlin, and Deepiga. \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Global variables

char current\_Player[25];

int end\_flag=0;//used in end\_game() function

int help\_flag=0;//used in help() function

int check\_flag\_X=0;//if it's set to 1, then move made by player 1 is invalid

int check\_flag\_O=0;//if it's set to 1, then move made by player 2 is invalid

int king\_flag=0; //for whoever king\_flag=1, that token is king

struct token{

int row, col;// y-axis and x-axis coordinate

};

struct player{

int count;//count for score

char name[25];//name of player

};

struct player player\_X, player\_O;

char checkers[8][8]={

{'O',' ','O',' ','O',' ','O',' '},

{' ','O',' ','O',' ','O',' ','O'},

{'O',' ','O',' ','O',' ','O',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ','X',' ','X',' ','X',' ','X'},

{'X',' ','X',' ','X',' ','X',' '},

{' ','X',' ','X',' ','X',' ','X'}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of declaration of Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* START of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++

void screen\_clear(void){

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ends game and program exits with exit code

int end\_game(struct token old){

//9 9 is surrender code

if(old.row==9 && old.col==9)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

if(strcmp(current\_Player,player\_X.name)==0)

{

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

else

{

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

}

//player 2 captured all X tokens, so game over

else if(player\_O.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

//player 1 captured all O tokens, so game over

else if(player\_X.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

return end\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of end\_game() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if king token or not

void check\_king()

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_king() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token X is valid or not

int check\_valid\_X(struct token old, struct token new)

{

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(checkers[new.row][new.col]!=checkers[old.row-1][old.col+1])

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Try again.\n");

}

//Anything other than one diagonally left movement not allowed

else if(checkers[new.row][new.col]!=checkers[old.row-1][old.col-1])

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Try again.\n");

}

//player 1 has X tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Try again.\n");

}

//Token can't be placed in a square where another X token is already present

else if(checkers[new.row][new.col]=='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Try again.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Try again.\n");

}

}

return check\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token O is valid or not

int check\_valid\_O(struct token old, struct token new)

{

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(checkers[new.row][new.col]!=checkers[old.row+1][old.col+1])

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Try again.\n");

}

//Anything other than one diagonally left movement not allowed

else if(checkers[new.row][new.col]!=checkers[old.row+1][old.col-1])

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Try again.\n");

}

//player 2 has O tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Try again.\n");

}

//Token can't be placed in a square where another O token is already present

else if(checkers[new.row][new.col]=='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Try again.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Try again.\n");

}

}

return check\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//8 8 is help code

int help(struct token old){

if(old.row==8 && old.col==8)

{

help\_flag=1;

yellow();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. Enter 9 9 as token coordinates to surrender.\n9. Enter 8 8 as token coordinates for help menu.");

printf("\n10. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

return help\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of help() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//coding for checker board and tokens using user-defined function

void draw\_board(void)

{

int i,j;

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

printf(" --- --- --- --- --- --- --- --- ");

for(i=0;i<8;i++)

{

printf(" \nr%d ",i);

for(j=0;j<8;j++)

{

printf("| %c ",checkers[i][j]);

}

printf("| r%d \n",i);

if(i==0||i==1||i==2||i==3||i==4||i==5||i==6||i==7)

printf(" --- --- --- --- --- --- --- --- ");

}

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of draw\_board() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_O(struct token old, struct token new)//for player2

{

//O token should move only if it's valid

if(check\_flag\_O==0)

{

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

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

}

printf("\n new checkers[%d][%d]=%c \n",new.row,new.col,checkers[new.row][new.col]);

printf("old checkers[%d][%d]=%c \n",old.row,old.col, checkers[old.row][old.col]);

strcpy(current\_Player,player\_O.name);

if(end\_game(old)==0)

{

green();

draw\_board();

}

if(checkers[new.row][new.col]=='X')

player\_O.count++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_X(struct token old, struct token new)//for player1

{

//X token should move only if it's valid

if(check\_flag\_X==0)

{

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

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

}

printf("\n new checkers[%d][%d]=%c \n",new.row,new.col,checkers[new.row][new.col]);

printf("old checkers[%d][%d]=%c \n",old.row,old.col, checkers[old.row][old.col]);

strcpy(current\_Player,player\_X.name);

if(end\_game(old)==0)

{

green();

draw\_board();

}

if(checkers[new.row][new.col]=='O')

player\_X.count++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_turn(int Turn, struct player player\_X, struct player player\_O){

struct token old, new;

int rem =0;//remainder of Turn%2

for(Turn=1; ;Turn++)

{

if(end\_flag==1)

break;

if(Turn%2==0)

{

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

//help(old);

help\_flag=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

move\_O(old, new);

}

if(check\_valid\_O(old,new))

{

//check\_flag\_O=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

//check\_valid\_O(old,new);

move\_O(old, new);

}

move\_O(old, new);

}

else

{

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

move\_X(old, new);

}

if(check\_valid\_X(old,new))

{

//check\_flag\_X=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

//check\_valid\_X(old,new);

move\_X(old, new);

}

move\_X(old, new);

rem = 1;

}

}

return rem;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of get\_turn() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* END of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of main() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

//char player1[25], player2[25];

int turn=2;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to Start Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

cyan();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DRAUGHTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter name of player 1:");

scanf("%s", player\_X.name);

printf("Enter name of player 2:");

scanf("%s", player\_O.name);

getchar();

screen\_clear();

red();

printf("\n\*\*\*Good day %s and %s, let's start the game. All the best!!!\*\*\*\n", player\_X.name, player\_O.name);

printf("\n");

blue();

printf("\n# Objective: Capture Opponent's token");

printf("\n# Cutting of opponent's token: A player may jump one of his tokens onto one square occupied by his opponent's token.");

printf("\n# Captured token: The opponent's token is cut by the player's token and removed from the board.");

printf("\n# Points: Number of tokens cut by the player is the player's score.\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

green();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. Enter 9 9 as token coordinates to surrender.\n9. Enter 8 8 as token coordinates for help menu.");

printf("\n10. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

draw\_board();

turn = get\_turn(turn, player\_X, player\_O);

return 0;

}

Progress #14

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to draught programed by Samuela, Sakthi, Merlin, and Deepiga. \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Global variables

char current\_Player[25];

int end\_flag=0;//used in end\_game() function

int help\_flag=0;//used in help() function

int check\_flag\_X=0;//if it's set to 1, then move made by player 1 is invalid

int check\_flag\_O=0;//if it's set to 1, then move made by player 2 is invalid

int king\_flag=0; //for whoever king\_flag=1, that token is king

struct token{

int row, col;// y-axis and x-axis coordinate

};

struct player{

int count;//count for score

char name[25];//name of player

};

struct player player\_X, player\_O;

char checkers[8][8]={

{'O',' ','O',' ','O',' ','O',' '},

{' ','O',' ','O',' ','O',' ','O'},

{'O',' ','O',' ','O',' ','O',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ','X',' ','X',' ','X',' ','X'},

{'X',' ','X',' ','X',' ','X',' '},

{' ','X',' ','X',' ','X',' ','X'}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of declaration of Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* START of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++

void screen\_clear(void){

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ends game and program exits with exit code

int end\_game(struct token old){

//9 9 is surrender code

if(old.row==9 && old.col==9)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

if(strcmp(current\_Player,player\_X.name)==0)

{

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

else

{

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

}

//player 2 captured all X tokens, so game over

else if(player\_O.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

//player 1 captured all O tokens, so game over

else if(player\_X.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

return end\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of end\_game() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if king token or not

void check\_king()

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_king() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token X is valid or not

int check\_valid\_X(struct token old, struct token new)

{

int n=0;

//X tokens shouldn't move vertically forward

for(n=0;n<8;n++)

{

if(king\_flag==0 && checkers[new.row][new.col]==checkers[old.row -n][new.col])

check\_flag\_X=1;

}

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(checkers[new.row][new.col]!=checkers[old.row-1][old.col+1])

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(checkers[new.row][new.col]!=checkers[old.row-1][old.col-1])

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//player 1 has X tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select any X token to move.\n");

}

//Token can't be placed in a square where another X token is already present

else if(checkers[old.row][old.col]=='X' && checkers[new.row][new.col]=='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select some other square to place token X.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not be placed outside checker board.\n");

}

}

return check\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token O is valid or not

int check\_valid\_O(struct token old, struct token new)

{

int n=0;

//O tokens shouldn't move vertically forward

for(n=0;n<8;n++)

{

if(king\_flag==0 && checkers[new.row][new.col]==checkers[old.row +n][new.col])

check\_flag\_O=1;

}

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(checkers[new.row][new.col]!=checkers[old.row+1][old.col+1])

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(checkers[new.row][new.col]!=checkers[old.row+1][old.col-1])

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//player 2 has O tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select any O token to move.\n");

}

//Token can't be placed in a square where another O token is already present

else if(checkers[old.row][old.col]=='O' && checkers[new.row][new.col]=='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select some other square to place token O.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not be placed outside checker board.\n");

}

}

return check\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//8 8 is help code

int help(struct token old){

if(old.row==8 && old.col==8)

{

help\_flag=1;

yellow();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

return help\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of help() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//coding for checker board and tokens using user-defined function

void draw\_board(void)

{

int i,j;

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

printf(" --- --- --- --- --- --- --- --- ");

for(i=0;i<8;i++)

{

printf(" \nr%d ",i);

for(j=0;j<8;j++)

{

printf("| %c ",checkers[i][j]);

}

printf("| r%d \n",i);

if(i==0||i==1||i==2||i==3||i==4||i==5||i==6||i==7)

printf(" --- --- --- --- --- --- --- --- ");

}

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of draw\_board() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_O(struct token old, struct token new)//for player2

{

//O token should move only if it's valid

if(check\_flag\_O==0)

{

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

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

}

getchar();//if getchar() isn't used, screen clears everything even before properly displaying content first

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_O.name);

if(end\_game(old)==0)

{

green();

draw\_board();

}

if(checkers[new.row][new.col]=='X')

player\_O.count++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_X(struct token old, struct token new)//for player1

{

//X token should move only if it's valid

if(check\_flag\_X==0)

{

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

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

}

getchar();

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_X.name);

if(end\_game(old)==0)

{

green();

draw\_board();

}

if(checkers[new.row][new.col]=='O')

player\_X.count++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_turn(int Turn, struct player player\_X, struct player player\_O){

struct token old, new;

int rem =0;//remainder of Turn%2

for(Turn=1; ;Turn++)

{

if(end\_flag==1)

break;

if(Turn%2==0)

{

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_O(old,new))

{

check\_flag\_O=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(check\_valid\_O(old,new))

{

check\_flag\_O=1;

}

}

move\_O(old, new);

}

else

{

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\nEnter 9 9 as token row number and column number to surrender.\n9. Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_X(old,new))

{

check\_flag\_X=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(check\_valid\_X(old,new))

{

check\_flag\_X=1;

}

}

move\_X(old, new);

rem = 1;

}

}

return rem;//rem will determine turn while calling this function in main() function

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of get\_turn() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* END of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of main() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

int turn=2;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to Start Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

cyan();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DRAUGHTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter name of player 1:");

scanf("%s", player\_X.name);

printf("Enter name of player 2:");

scanf("%s", player\_O.name);

getchar();

screen\_clear();

red();

printf("\n\*\*\*Good day %s and %s, let's start the game. All the best!!!\*\*\*\n", player\_X.name, player\_O.name);

printf("\n");

blue();

printf("\n# Objective: Capture Opponent's token");

printf("\n# Cutting of opponent's token: A player may jump one of his tokens onto one square occupied by his opponent's token.");

printf("\n# Captured token: The opponent's token is cut by the player's token and removed from the board.");

printf("\n# Points: Number of tokens cut by the player is the player's score.\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

green();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. Enter 9 9 as token row number and column number to surrender.\n9. Enter 8 8 as token row number and column number for help menu.");

printf("\n10. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

draw\_board();

turn = get\_turn(turn, player\_X, player\_O);

return 0;

}

Progress #15

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to draught programed by Samuela Abigail- 71762108039, \*

\* Sakthi Abinaya S- 71762108038, \*

\* Merlin Might V S- 71762108027, \*

\* and Deepiga V- 71762108007. \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\* \*

\* C File name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.c \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Global variables

char current\_Player[25];

int end\_flag=0;//used in end\_game() function

int help\_flag=0;//used in help() function

int check\_flag\_X=0;//if it's set to 1, then move made by player 1 is invalid

int check\_flag\_O=0;//if it's set to 1, then move made by player 2 is invalid

int king\_flag=0; //for whoever king\_flag=1, that token is king

struct token{

int row, col;// y-axis and x-axis coordinate

};

struct player{

int count;//count for score

char name[25];//name of player

};

struct player player\_X, player\_O;

char checkers[8][8]={

{'O',' ','O',' ','O',' ','O',' '},

{' ','O',' ','O',' ','O',' ','O'},

{'O',' ','O',' ','O',' ','O',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ','X',' ','X',' ','X',' ','X'},

{'X',' ','X',' ','X',' ','X',' '},

{' ','X',' ','X',' ','X',' ','X'}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of declaration of Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* START of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear(void){

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ends game and program exits with exit code

int end\_game(struct token old){

//9 9 is surrender code

if(old.row==9 && old.col==9)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

if(strcmp(current\_Player,player\_X.name)==0)

{

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

else

{

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

}

//player 2 captured all X tokens, so game over

else if(player\_O.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

//player 1 captured all O tokens, so game over

else if(player\_X.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

return end\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of end\_game() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if king token or not

void check\_king()

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_king() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token X is valid or not

int check\_valid\_X(struct token old, struct token new)

{

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row-1) && new.col!=(old.col+1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row-1) && new.col!=(old.col-1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//player 1 has X tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select any X token to move.\n");

}

//Token can't be placed in a square where another X token is already present

else if(checkers[new.row][new.col]=='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select some other square to place token X.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not be placed outside checker board.\n");

}

}

//in case of help menu in 2nd chance

if(old.row==8 && old.col==8)

{

check\_flag\_X=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_X=0;

}

return check\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token O is valid or not

int check\_valid\_O(struct token old, struct token new)

{

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row+1) && new.col!=(old.col+1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row+1) && new.col!=(old.col-1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//player 2 has O tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select any O token to move.\n");

}

//Token can't be placed in a square where another O token is already present

else if(checkers[new.row][new.col]=='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select some other square to place token O.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not be placed outside checker board.\n");

}

}

//in case of help menu in 2nd chance

if(old.row==8 && old.col==8)

{

check\_flag\_O=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_O=0;

}

return check\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//8 8 is help code

int help(struct token old){

if(old.row==8 && old.col==8)

{

help\_flag=1;

yellow();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

return help\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of help() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//coding for checker board and tokens using user-defined function

void draw\_board(void)

{

int i,j;

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

printf(" --- --- --- --- --- --- --- --- ");

for(i=0;i<8;i++)

{

printf(" \nr%d ",i);

for(j=0;j<8;j++)

{

printf("| %c ",checkers[i][j]);

}

printf("| r%d \n",i);

if(i==0||i==1||i==2||i==3||i==4||i==5||i==6||i==7)

printf(" --- --- --- --- --- --- --- --- ");

}

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of draw\_board() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_O(struct token old, struct token new)//for player2

{

//again checking in case of 2nd chance

check\_valid\_O(old,new);

//O token should move only if it's valid

if(check\_flag\_O==0)

{

if(checkers[new.row][new.col]=='X')

{

player\_O.count++;//every time O token cuts X, count will be incremented by 1

}

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

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

}

getchar();//if getchar() isn't used, screen clears everything even before properly displaying content first

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_O.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_X(struct token old, struct token new)//for player1

{

//again checking in case of 2nd chance

check\_valid\_X(old,new);

//X token should move only if it's valid

if(check\_flag\_X==0)

{

if(checkers[new.row][new.col]=='O')

{

player\_X.count++;//every time X token cuts O, count will be incremented by 1

}

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

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

}

getchar();

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_X.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_turn(int Turn, struct player player\_X, struct player player\_O){

struct token old, new;

int rem =0;//remainder of Turn%2

for(Turn=1; ;Turn++)

{

if(end\_flag==1)

break;

if(Turn%2==0)

{

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_O(old,new))

{

check\_flag\_O=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_O(old, new);

}

else

{

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\nEnter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_X(old,new))

{

check\_flag\_X=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_X(old, new);

rem = 1;

}

}

return rem;//rem will determine turn while calling this function in main() function

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of get\_turn() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* END of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of main() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

int turn=2;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to Start Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

cyan();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DRAUGHTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter name of player 1:");

scanf("%s", player\_X.name);

printf("Enter name of player 2:");

scanf("%s", player\_O.name);

getchar();

screen\_clear();

red();

printf("\n\*\*\*Good day %s and %s, let's start the game. All the best!!!\*\*\*\n", player\_X.name, player\_O.name);

printf("\n");

blue();

printf("\n# Objective: Capture Opponent's token");

printf("\n# Cutting of opponent's token: A player may jump one of his tokens onto one square occupied by his opponent's token.");

printf("\n# Captured token: The opponent's token is cut by the player's token and removed from the board.");

printf("\n# Points: Number of tokens cut by the player is the player's score.\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

green();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. Enter 9 9 as token row number and column number to surrender.\n9. Enter 8 8 as token row number and column number for help menu.");

printf("\n10. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

draw\_board();

turn = get\_turn(turn, player\_X, player\_O);

return 0;

}

Progress #16 (has fool-proof check valid functions as far as I’ve tested. Only bug is that it won’t call help function in case of 2nd chance turn)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to draught programed by Samuela Abigail- 71762108039, \*

\* Sakthi Abinaya S- 71762108038, \*

\* Merlin Might V S- 71762108027, \*

\* and Deepiga V- 71762108007. \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\* \*

\* C File name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.c \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Global variables

char current\_Player[25];

int end\_flag=0;//used in end\_game() function

int help\_flag=0;//used in help() function

int check\_flag\_X=0;//if it's set to 1, then move made by player 1 is invalid

int check\_flag\_O=0;//if it's set to 1, then move made by player 2 is invalid

int king\_flag=0; //for whoever king\_flag=1, that token is king

struct token{

int row, col;// y-axis and x-axis coordinate

};

struct player{

int count;//count for score

char name[25];//name of player

};

struct player player\_X, player\_O;

char checkers[8][8]={

{'O',' ','O',' ','O',' ','O',' '},

{' ','O',' ','O',' ','O',' ','O'},

{'O',' ','O',' ','O',' ','O',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ','X',' ','X',' ','X',' ','X'},

{'X',' ','X',' ','X',' ','X',' '},

{' ','X',' ','X',' ','X',' ','X'}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of declaration of Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* START of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear(void){

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ends game and program exits with exit code

int end\_game(struct token old){

//9 9 is surrender code

if(old.row==9 && old.col==9)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

if(strcmp(current\_Player,player\_X.name)==0)

{

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

else

{

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

}

//player 2 captured all X tokens, so game over

else if(player\_O.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

//player 1 captured all O tokens, so game over

else if(player\_X.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

return end\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of end\_game() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if king token or not

void check\_king()

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_king() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token X is valid or not

int check\_valid\_X(struct token old, struct token new)

{

int n;

//It's jumping n columns right despite above rules, so again specifying

for(n=2,king\_flag=0; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_X=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2,king\_flag=0; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_X=1;

}

}

if(check\_flag\_X==1 && king\_flag==0)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token X should not jump columns.\n");

}

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row-1) && new.col!=(old.col+1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row-1) && new.col!=(old.col-1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//player 1 has X tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select any X token to move.\n");

}

//Token can't be placed in a square where another X token is already present

else if(checkers[new.row][new.col]=='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select some other square to place token X.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not be placed outside checker board.\n");

}

}

//in case of help menu in 2nd chance

if(old.row==8 && old.col==8)

{

check\_flag\_X=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_X=0;

}

return check\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token O is valid or not

int check\_valid\_O(struct token old, struct token new)

{

int n;

//It's jumping n columns right despite above rules, so again specifying

for(n=2,king\_flag=0; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_O=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2,king\_flag=0; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_O=1;

}

}

if(check\_flag\_O==1 && king\_flag==0)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token O should not jump columns.\n");

}

if(king\_flag==0)

{

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row+1) && new.col!=(old.col+1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row+1) && new.col!=(old.col-1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//It's moving vertically upwards into next row despite above rules, so again specifying

if(new.row==(old.row+1) && new.col==old.col )

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not move vertically upwards.\n");

}

//player 2 has O tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select any O token to move.\n");

}

//Token can't be placed in a square where another O token is already present

else if(checkers[new.row][new.col]=='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select some other square to place token O.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not be placed outside checker board.\n");

}

}

//in case of help menu in 2nd chance

if(old.row==8 && old.col==8)

{

check\_flag\_O=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_O=0;

}

return check\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//8 8 is help code

int help(struct token old){

if(old.row==8 && old.col==8)

{

help\_flag=1;

yellow();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

return help\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of help() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//coding for checker board and tokens using user-defined function

void draw\_board(void)

{

int i,j;

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

printf(" --- --- --- --- --- --- --- --- ");

for(i=0;i<8;i++)

{

printf(" \nr%d ",i);

for(j=0;j<8;j++)

{

printf("| %c ",checkers[i][j]);

}

printf("| r%d \n",i);

if(i==0||i==1||i==2||i==3||i==4||i==5||i==6||i==7)

printf(" --- --- --- --- --- --- --- --- ");

}

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of draw\_board() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_O(struct token old, struct token new)//for player2

{

//again checking in case of 2nd chance

check\_valid\_O(old,new);

//O token should move only if it's valid

if(check\_flag\_O==0)

{

if(checkers[new.row][new.col]=='X')

{

player\_O.count++;//every time O token cuts X, count will be incremented by 1

}

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

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

}

getchar();//if getchar() isn't used, screen clears everything even before properly displaying content first

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_O.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_X(struct token old, struct token new)//for player1

{

//again checking in case of 2nd chance

check\_valid\_X(old,new);

//X token should move only if it's valid

if(check\_flag\_X==0)

{

if(checkers[new.row][new.col]=='O')

{

player\_X.count++;//every time X token cuts O, count will be incremented by 1

}

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

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

}

getchar();

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_X.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_turn(int Turn, struct player player\_X, struct player player\_O){

struct token old, new;

int rem =0;//remainder of Turn%2

for(Turn=1; ;Turn++)

{

if(end\_flag==1)

break;

if(Turn%2==0)

{

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_O(old,new))

{

check\_flag\_O=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_O(old, new);

}

else

{

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_X(old,new))

{

check\_flag\_X=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_X(old, new);

rem = 1;

}

}

return rem;//rem will determine turn while calling this function in main() function

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of get\_turn() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* END of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of main() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

int turn=2;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to Start Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

cyan();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DRAUGHTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter name of player 1:");

scanf("%s", player\_X.name);

printf("Enter name of player 2:");

scanf("%s", player\_O.name);

getchar();

screen\_clear();

red();

printf("\n\*\*\*Good day %s and %s, let's start the game. All the best!!!\*\*\*\n", player\_X.name, player\_O.name);

printf("\n");

blue();

printf("\n# Objective: Capture Opponent's token");

printf("\n# Cutting of opponent's token: A player may jump one of his tokens onto one square occupied by his opponent's token.");

printf("\n# Captured token: The opponent's token is cut by the player's token and removed from the board.");

printf("\n# Points: Number of tokens cut by the player is the player's score.\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

green();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. Enter 9 9 as token row number and column number to surrender.\n9. Enter 8 8 as token row number and column number for help menu.");

printf("\n10. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

draw\_board();

turn = get\_turn(turn, player\_X, player\_O);

return 0;

}

Progress #17 (has non-working king() function)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to draught programed by Samuela Abigail- 71762108039, \*

\* Sakthi Abinaya S- 71762108038, \*

\* Merlin Might V S- 71762108027, \*

\* and Deepiga V- 71762108007. \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\* \*

\* C File name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.c \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Global variables

char current\_Player[25];

int end\_flag=0;//used in end\_game() function

int help\_flag=0;//used in help() function

int check\_flag\_X=0;//if it's set to 1, then move made by player 1 is invalid

int check\_flag\_O=0;//if it's set to 1, then move made by player 2 is invalid

int king\_flag\_X=0; //for whoever king\_flag\_X=1, that X token is king

int king\_flag\_O=0; //for whoever king\_flag\_O=1, that O token is king

struct token{

int row, col;// y-axis and x-axis coordinate respectively

};

struct player{

int count;//count for score

char name[25];//name of player

};

struct player player\_X, player\_O;

char checkers[8][8]={

{'O',' ','O',' ','O',' ','O',' '},

{' ','O',' ','O',' ','O',' ','O'},

{'O',' ','O',' ','O',' ','O',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ','X',' ','X',' ','X',' ','X'},

{'X',' ','X',' ','X',' ','X',' '},

{' ','X',' ','X',' ','X',' ','X'}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of declaration of Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* START of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear(void){

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ends game and program exits with exit code

int end\_game(struct token old){

//9 9 is surrender code

if(old.row==9 && old.col==9)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

if(strcmp(current\_Player,player\_X.name)==0)

{

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

else

{

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

}

//player 2 captured all X tokens, so game over

else if(player\_O.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

//player 1 captured all O tokens, so game over

else if(player\_X.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

return end\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of end\_game() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if king token or not for X

int check\_king\_X(struct token old, struct token new)

{

if(new.row==0)//X token has reached other end of board

{

king\_flag\_X=1;

red();//to mark token in red color

checkers[old.row][old.col];//it hasn't gone into move\_X function yet, so 'new' can't be used

}

return king\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_king\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if king token or not for O

int check\_king\_O(struct token old, struct token new)

{

if(new.row==7)//O token has reached other end of board

{

king\_flag\_O=1;

red();//to mark token in red color

checkers[old.row][old.col];//it hasn't gone into move\_O function yet, so 'new' can't be used

}

return king\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_king\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token X is valid or not

int check\_valid\_X(struct token old, struct token new)

{

int n;

check\_king\_X(old,new);//calling function to check king\_flag\_X

//It's jumping n columns right despite below rules, so again specifying

for(n=2,king\_flag\_X=0; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_X=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2,king\_flag\_X=0; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_X=1;

}

}

if(check\_flag\_X==1 && king\_flag\_X==0)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token X should not jump columns.\n");

}

if(king\_flag\_X==0)

{

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row-1) && new.col!=(old.col+1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row-1) && new.col!=(old.col-1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//It's moving vertically upwards into next row despite above rules, so again specifying

if(new.row==(old.row-1) && new.col==old.col )

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not move vertically forward.\n");

}

//player 1 has X tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select any X token to move.\n");

}

//Token can't be placed in a square where another X token is already present

else if(checkers[new.row][new.col]=='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select some other square to place token X.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not be placed outside checker board.\n");

}

}

//in case of help menu in 2nd chance

if(old.row==8 && old.col==8)

{

check\_flag\_X=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_X=0;

}

return check\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token O is valid or not

int check\_valid\_O(struct token old, struct token new)

{

int n;

check\_king\_O(old,new);//calling function to check king\_flag\_O

//It's jumping n columns right despite below rules, so again specifying

for(n=2,king\_flag\_O=0; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_O=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2,king\_flag\_O=0; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_O=1;

}

}

if(check\_flag\_O==1 && king\_flag\_O==0)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token O should not jump columns.\n");

}

if(king\_flag\_O==0)

{

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row+1) && new.col!=(old.col+1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row+1) && new.col!=(old.col-1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//It's moving vertically upwards into next row despite above rules, so again specifying

if(new.row==(old.row+1) && new.col==old.col )

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not move vertically forward.\n");

}

//player 2 has O tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select any O token to move.\n");

}

//Token can't be placed in a square where another O token is already present

else if(checkers[new.row][new.col]=='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select some other square to place token O.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not be placed outside checker board.\n");

}

}

//in case of help menu in 2nd chance

if(old.row==8 && old.col==8)

{

check\_flag\_O=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_O=0;

}

return check\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//8 8 is help code

int help(struct token old){

if(old.row==8 && old.col==8)

{

help\_flag=1;

yellow();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

return help\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of help() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//coding for checker board and tokens using user-defined function

void draw\_board(void)

{

int i,j;

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

printf(" --- --- --- --- --- --- --- --- ");

for(i=0;i<8;i++)

{

printf(" \nr%d ",i);

for(j=0;j<8;j++)

{

printf("| %c ",checkers[i][j]);

}

printf("| r%d \n",i);

if(i==0||i==1||i==2||i==3||i==4||i==5||i==6||i==7)

printf(" --- --- --- --- --- --- --- --- ");

}

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of draw\_board() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_O(struct token old, struct token new)//for player2

{

//again checking in case of 2nd chance

check\_valid\_O(old,new);

//O token should move only if it's valid

if(check\_flag\_O==0)

{

if(checkers[new.row][new.col]=='X')

{

player\_O.count++;//every time O token cuts X, count will be incremented by 1

}

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

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

}

getchar();//if getchar() isn't used, screen clears everything even before properly displaying content first

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_O.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_X(struct token old, struct token new)//for player1

{

//again checking in case of 2nd chance

check\_valid\_X(old,new);

//X token should move only if it's valid

if(check\_flag\_X==0)

{

if(checkers[new.row][new.col]=='O')

{

player\_X.count++;//every time X token cuts O, count will be incremented by 1

}

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

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

}

getchar();

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_X.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_turn(int Turn, struct player player\_X, struct player player\_O){

struct token old, new;

int rem =0;//remainder of Turn%2

for(Turn=1; ;Turn++)

{

if(end\_flag==1)

break;

if(Turn%2==0)

{

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_O(old,new))

{

check\_flag\_O=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_O(old, new);

}

else

{

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_X(old,new))

{

check\_flag\_X=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_X(old, new);

rem = 1;

}

}

return rem;//rem will determine turn while calling this function in main() function

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of get\_turn() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* END of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of main() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

int turn=2;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to Start Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

cyan();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DRAUGHTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter name of player 1:");

scanf("%s", player\_X.name);

printf("Enter name of player 2:");

scanf("%s", player\_O.name);

getchar();

screen\_clear();

red();

printf("\n\*\*\*Good day %s and %s, let's start the game. All the best!!!\*\*\*\n", player\_X.name, player\_O.name);

printf("\n");

blue();

printf("\n# Objective: Capture Opponent's token");

printf("\n# Cutting of opponent's token: A player may jump one of his tokens onto one square occupied by his opponent's token.");

printf("\n# Captured token: The opponent's token is cut by the player's token and removed from the board.");

printf("\n# Points: Number of tokens cut by the player is the player's score.\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

green();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n3. Once a token reaches other end of the board, it'll become king and can move in all directions,\n any number of squares during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n6. Each Player has 12 tokens on a 8x8 square board.");

printf("\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.");

printf("\n8. Enter 9 9 as token row number and column number to surrender.\n9. Enter 8 8 as token row number and column number for help menu.");

printf("\n10. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

draw\_board();

turn = get\_turn(turn, player\_X, player\_O);

return 0;

}

Progress #18 (removed king() function)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to draught programed by Samuela Abigail- 71762108039, \*

\* Sakthi Abinaya S- 71762108038, \*

\* Merlin Might V S- 71762108027, \*

\* and Deepiga V- 71762108007. \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\* \*

\* C File name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.c \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Global variables

char current\_Player[25];

int end\_flag=0;//used in end\_game() function

int help\_flag=0;//used in help() function

int check\_flag\_X=0;//if it's set to 1, then move made by player 1 is invalid

int check\_flag\_O=0;//if it's set to 1, then move made by player 2 is invalid

struct token{

int row, col;// y-axis and x-axis coordinate respectively

};

struct player{

int count;//count for score

char name[25];//name of player

};

struct player player\_X, player\_O;

char checkers[8][8]={

{'O',' ','O',' ','O',' ','O',' '},

{' ','O',' ','O',' ','O',' ','O'},

{'O',' ','O',' ','O',' ','O',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ','X',' ','X',' ','X',' ','X'},

{'X',' ','X',' ','X',' ','X',' '},

{' ','X',' ','X',' ','X',' ','X'}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of declaration of Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* START of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear(void){

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ends game and program exits with exit code

int end\_game(struct token old){

//9 9 is surrender code

if(old.row==9 && old.col==9)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

if(strcmp(current\_Player,player\_X.name)==0)

{

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

else

{

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

}

//player 2 captured all X tokens, so game over

else if(player\_O.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

//player 1 captured all O tokens, so game over

else if(player\_X.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

return end\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of end\_game() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token X is valid or not

int check\_valid\_X(struct token old, struct token new)

{

int n;

//It's jumping n columns right despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_X=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_X=1;

}

}

if(check\_flag\_X==1)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token X should not jump columns.\n");

}

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row-1) && new.col!=(old.col+1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row-1) && new.col!=(old.col-1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//It's moving vertically upwards into next row despite above rules, so again specifying

if(new.row==(old.row-1) && new.col==old.col )

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not move vertically forward.\n");

}

//player 1 has X tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select any X token to move.\n");

}

//Token can't be placed in a square where another X token is already present

else if(checkers[new.row][new.col]=='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select some other square to place token X.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not be placed outside checker board.\n");

}

//in case of help menu in 2nd chance

else if(old.row==8 && old.col==8)

{

check\_flag\_X=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_X=0;

}

return check\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token O is valid or not

int check\_valid\_O(struct token old, struct token new)

{

int n;

//It's jumping n columns right despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_O=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_O=1;

}

}

if(check\_flag\_O==1)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token O should not jump columns.\n");

}

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row+1) && new.col!=(old.col+1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row+1) && new.col!=(old.col-1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//It's moving vertically upwards into next row despite above rules, so again specifying

else if(new.row==(old.row+1) && new.col==old.col )

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not move vertically forward.\n");

}

//player 2 has O tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select any O token to move.\n");

}

//Token can't be placed in a square where another O token is already present

else if(checkers[new.row][new.col]=='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select some other square to place token O.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not be placed outside checker board.\n");

}

//in case of help menu in 2nd chance

else if(old.row==8 && old.col==8)

{

check\_flag\_O=0;

}

//in case of surrender in 2nd chance

else if(old.row==9 && old.col==9)

{

check\_flag\_O=0;

}

return check\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//8 8 is help code

int help(struct token old){

if(old.row==8 && old.col==8)

{

help\_flag=1;

yellow();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n5. Each Player has 12 tokens on a 8x8 square board.");

printf("\n6. 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.");

printf("\n7. Enter 9 9 as token row number and column number to surrender.\n8. Enter 8 8 as token row number and column number for help menu.");

printf("\n9. 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");

}

return help\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of help() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//coding for checker board and tokens using user-defined function

void draw\_board(void)

{

int i,j;

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

printf(" --- --- --- --- --- --- --- --- ");

for(i=0;i<8;i++)

{

printf(" \nr%d ",i);

for(j=0;j<8;j++)

{

printf("| %c ",checkers[i][j]);

}

printf("| r%d \n",i);

if(i==0||i==1||i==2||i==3||i==4||i==5||i==6||i==7)

printf(" --- --- --- --- --- --- --- --- ");

}

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of draw\_board() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_O(struct token old, struct token new)//for player2

{

//again checking in case of 2nd chance

check\_valid\_O(old,new);

//O token should move only if it's valid

if(check\_flag\_O==0)

{

if(checkers[new.row][new.col]=='X')

{

player\_O.count++;//every time O token cuts X, count will be incremented by 1

}

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

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

}

getchar();//if getchar() isn't used, screen clears everything even before properly displaying content first

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_O.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_X(struct token old, struct token new)//for player1

{

//again checking in case of 2nd chance

check\_valid\_X(old,new);

//X token should move only if it's valid

if(check\_flag\_X==0)

{

if(checkers[new.row][new.col]=='O')

{

player\_X.count++;//every time X token cuts O, count will be incremented by 1

}

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

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

}

getchar();

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_X.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_turn(int Turn, struct player player\_X, struct player player\_O){

struct token old, new;

int rem =0;//remainder of Turn%2

for(Turn=1; ;Turn++)

{

if(end\_flag==1)

break;

if(Turn%2==0)

{

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_O(old,new))

{

check\_flag\_O=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_O(old, new);

}

else

{

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))

{

help\_flag=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_X(old,new))

{

check\_flag\_X=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

move\_X(old, new);

rem = 1;

}

}

return rem;//rem will determine turn while calling this function in main() function

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of get\_turn() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* END of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of main() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

int turn=2;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to Start Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

cyan();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DRAUGHTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter name of player 1:");

scanf("%s", player\_X.name);

printf("Enter name of player 2:");

scanf("%s", player\_O.name);

getchar();

screen\_clear();

red();

printf("\n\*\*\*Good day %s and %s, let's start the game. All the best!!!\*\*\*\n", player\_X.name, player\_O.name);

printf("\n");

blue();

printf("\n# Objective: Capture Opponent's token");

printf("\n# Cutting of opponent's token: A player may jump one of his tokens onto one square occupied by his opponent's token.");

printf("\n# Captured token: The opponent's token is cut by the player's token and removed from the board.");

printf("\n# Points: Number of tokens cut by the player is the player's score.\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

green();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n5. Each Player has 12 tokens on a 8x8 square board.");

printf("\n6. 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.");

printf("\n7. Enter 9 9 as token row number and column number to surrender.\n8. Enter 8 8 as token row number and column number for help menu.");

printf("\n9. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

draw\_board();

turn = get\_turn(turn, player\_X, player\_O);

return 0;

}

Progress #19 (latest fool-proof bug free program as far as tested, doesn’t have king() function)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Welcome to draught programed by Samuela Abigail- 71762108039, \*

\* Sakthi Abinaya S- 71762108038, \*

\* Merlin Might V S- 71762108027, \*

\* and Deepiga V- 71762108007. \*

\* \*

\* \*

\* AI & DS Department \*

\* Coimbatore Institute of Technology \*

\* \*

\* C File name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.c \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Global variables

char current\_Player[25];

int end\_flag=0;//used in end\_game() function

int help\_flag=0;//used in help() function

int check\_flag\_X=0;//if it's set to 1, then move made by player 1 is invalid

int check\_flag\_O=0;//if it's set to 1, then move made by player 2 is invalid

struct token{

int row, col;// y-axis and x-axis coordinate respectively

};

struct player{

int count;//count for score

char name[25];//name of player

};

struct player player\_X, player\_O;

char checkers[8][8]={

{'O',' ','O',' ','O',' ','O',' '},

{' ','O',' ','O',' ','O',' ','O'},

{'O',' ','O',' ','O',' ','O',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ',' ',' '},

{' ','X',' ','X',' ','X',' ','X'},

{'X',' ','X',' ','X',' ','X',' '},

{' ','X',' ','X',' ','X',' ','X'}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of declaration of Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//red color

void red(){

printf("\033[1;31m");

}

void reset();

//green color

void green(){

printf("\033[1;32m");

}

void reset();

//yellow color

void yellow(){

printf("\033[1;33m");

}

void reset();

//blue color

void blue(){

printf("\033[1;34m");

}

void reset();

//purple color

void purple(){

printf("\033[1;35m");

}

void reset();

//cyan color

void cyan(){

printf("\033[1;36m");

}

void reset();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of colored text functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* START of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//function to clear screen. Use cls for TurboC and clear for GCC/G++ compilers

void screen\_clear(void){

system("cls");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of screen\_clear() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ends game and program exits with exit code

int end\_game(struct token old){

//9 9 is surrender code

if(old.row==9 && old.col==9)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

if(strcmp(current\_Player,player\_X.name)==0)

{

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

else

{

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

}

//player 2 captured all X tokens, so game over

else if(player\_O.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_O.name);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

}

//player 1 captured all O tokens, so game over

else if(player\_X.count==12)

{

end\_flag=1;

screen\_clear();

red();

printf("\nGAME OVER!\n");

cyan();

printf("\n%s WON!\n",player\_X.name);

printf("\nScore of %s is %d\n",player\_X.name, player\_X.count);

printf("\nScore of %s is %d\n",player\_O.name, player\_O.count);

}

return end\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of end\_game() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token X is valid or not

int check\_valid\_X(struct token old, struct token new)

{

int n;

//in case of surrender in 2nd chance

if(old.row==9 && old.col==9)

{

end\_game(old);

}

//It's jumping n columns right despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_X=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_X=1;

}

}

if(check\_flag\_X==1)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token X should not jump columns.\n");

}

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row-1) && new.col!=(old.col+1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row-1) && new.col!=(old.col-1))

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should move diagonally one square in forward direction only.\n");

}

//It's moving vertically upwards into next row despite above rules, so again specifying

if(new.row==(old.row-1) && new.col==old.col )

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not move vertically forward.\n");

}

//player 1 has X tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select any X token to move.\n");

}

//Token can't be placed in a square where another X token is already present

else if(checkers[new.row][new.col]=='X')

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Select some other square to place token X.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_X=1;

red();

printf("\nInvalid move! Token X should not be placed outside checker board.\n");

}

//in case of help menu in 2nd chance

else if(old.row==8 && old.col==8)

{

check\_flag\_X=0;

}

return check\_flag\_X;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//check if move made by token O is valid or not

int check\_valid\_O(struct token old, struct token new)

{

int n;

//in case of surrender in 2nd chance

if(old.row==9 && old.col==9)

{

end\_game(old);

}

//It's jumping n columns right despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col+n))

{

check\_flag\_O=1;

}

}

//It's jumping n columns left despite below rules, so again specifying

for(n=2; n<8 ;n++)

{

if(new.col==(old.col-n))

{

check\_flag\_O=1;

}

}

if(check\_flag\_O==1)//checking flags of above for loops

{

red();

printf("\nInvalid move! Token O should not jump columns.\n");

}

//Anything other than one diagonally right movement not allowed

if(new.row!=(old.row+1) && new.col!=(old.col+1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//Anything other than one diagonally left movement not allowed

else if(new.row!=(old.row+1) && new.col!=(old.col-1))

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should move diagonally one square in forward direction only.\n");

}

//It's moving vertically upwards into next row despite above rules, so again specifying

else if(new.row==(old.row+1) && new.col==old.col )

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not move vertically forward.\n");

}

//player 2 has O tokens, so he can't select anything else

else if(checkers[old.row][old.col]!='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select any O token to move.\n");

}

//Token can't be placed in a square where another O token is already present

else if(checkers[new.row][new.col]=='O')

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Select some other square to place token O.\n");

}

//Token can't be placed outside the board

else if(new.row>7 && new.col>7)

{

check\_flag\_O=1;

red();

printf("\nInvalid move! Token O should not be placed outside checker board.\n");

}

//in case of help menu in 2nd chance

else if(old.row==8 && old.col==8)

{

check\_flag\_O=0;

}

return check\_flag\_O;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of check\_valid\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//8 8 is help code

int help(struct token old){

if(old.row==8 && old.col==8)

{

help\_flag=1;

yellow();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n5. Each Player has 12 tokens on a 8x8 square board.");

printf("\n6. 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.");

printf("\n7. Enter 9 9 as token row number and column number to surrender.\n8. Enter 8 8 as token row number and column number for help menu.");

printf("\n9. 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");

}

return help\_flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of help() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//coding for checker board and tokens using user-defined function

void draw\_board(void)

{

int i,j;

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

printf(" --- --- --- --- --- --- --- --- ");

for(i=0;i<8;i++)

{

printf(" \nr%d ",i);

for(j=0;j<8;j++)

{

printf("| %c ",checkers[i][j]);

}

printf("| r%d \n",i);

if(i==0||i==1||i==2||i==3||i==4||i==5||i==6||i==7)

printf(" --- --- --- --- --- --- --- --- ");

}

printf("\n c0 c1 c2 c3 c4 c5 c6 c7 \n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of draw\_board() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_O(struct token old, struct token new)//for player2

{

//again checking in case of 2nd chance

check\_valid\_O(old,new);

//O token should move only if it's valid

if(check\_flag\_O==0)

{

if(checkers[new.row][new.col]=='X')

{

player\_O.count++;//every time O token cuts X, count will be incremented by 1

}

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

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

}

getchar();//if getchar() isn't used, screen clears everything even before properly displaying content first

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_O.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_O() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void move\_X(struct token old, struct token new)//for player1

{

//again checking in case of 2nd chance

check\_valid\_X(old,new);

//X token should move only if it's valid

if(check\_flag\_X==0)

{

if(checkers[new.row][new.col]=='O')

{

player\_X.count++;//every time X token cuts O, count will be incremented by 1

}

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

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

}

getchar();

screen\_clear();//clears screen and draws updated board

strcpy(current\_Player,player\_X.name);

if(end\_game(old)==0)

{

green();

draw\_board();//draws updated board for next turn

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of move\_X() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_turn(int Turn, struct player player\_X, struct player player\_O){

struct token old, new;

int rem =0;//remainder of Turn%2

for(Turn=1; ;Turn++)

{

if(end\_flag==1)

break;

if(Turn%2==0)

{

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))//if player O entered 8 8

{

help\_flag=0;

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_O(old,new))//otherwise check if move is valid

{

if(end\_flag!=1)

{

check\_flag\_O=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (O's Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

}

move\_O(old, new);

}

else

{

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

if(help(old))//if player X entered 8 8

{

help\_flag=0;

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

if(check\_valid\_X(old,new))//otherwise check if move is valid

{

if(end\_flag!=1)

{

check\_flag\_X=0;//if not reset to zero means it will remain as 1 for that turn and won't move token even if valid move is made in 2nd chance turn

purple();

printf("\n %s's Turn (X's Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\n Enter 8 8 as token row number and column number for help menu.\n");

printf("\n Write row number and column number of token to be moved: ");

scanf("%d %d" ,&old.row ,&old.col);

printf("\n Write row number and column number of square where token is to be placed: ");

scanf("%d %d" ,&new.row ,&new.col);

}

}

move\_X(old, new);

rem = 1;

}

}

return rem;//rem will determine turn while calling this function in main() function

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* End of get\_turn() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* END of IMPORTANT game functions \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Start of main() function \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

int turn=2;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to Start Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

cyan();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DRAUGHTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter name of player 1:");

scanf("%s", player\_X.name);

printf("Enter name of player 2:");

scanf("%s", player\_O.name);

getchar();

screen\_clear();

red();

printf("\n\*\*\*Good day %s and %s, let's start the game. All the best!!!\*\*\*\n", player\_X.name, player\_O.name);

printf("\n");

blue();

printf("\n# Objective: Capture Opponent's token");

printf("\n# Cutting of opponent's token: A player may jump one of his tokens onto one square occupied by his opponent's token.");

printf("\n# Captured token: The opponent's token is cut by the player's token and removed from the board.");

printf("\n# Points: Number of tokens cut by the player is the player's score.\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

green();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RULES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please READ carefully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Tokens move only along the diagonal squares in forward direction.");

printf("\n2. Tokens move only one square during each turn.");

printf("\n4. You can continue the game until you capture all opponent tokens or until you surrender.");

printf("\n5. Player 1's token is 'X', Player 2's token is 'O'.\n5. Each Player has 12 tokens on a 8x8 square board.");

printf("\n6. 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.");

printf("\n7. Enter 9 9 as token row number and column number to surrender.\n8. Enter 8 8 as token row number and column number for help menu.");

printf("\n9. 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");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Press ENTER key to continue\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

screen\_clear();

draw\_board();

turn = get\_turn(turn, player\_X, player\_O);

return 0;

}

BIBLIOGRAPY

<https://cboard.cprogramming.com/c-programming/177496-checkers-game-finally-finished.html>

https://www.youtube.com/watch?v=Z\_f6yCvAZfQ