Progress #20

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

\* 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 \*

\* Notepad name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.txt \*

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

#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; //if king\_flag\_X=1, all X tokens will become king for 1 bonus turn

int king\_flag\_O=0; //if king\_flag\_O=1, all O tokens will become king for 1 bonus turn

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 \*

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

//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 player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.");

printf("\n He can move his token in all directions and any number of squares during bonus turn.\n But if he gets king in 2nd chance turn, bonus turn won't be given.");

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");

}

return help\_flag;

}

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

\* End of help() function \*

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

/\*\*\*\*if token becomes king, it can move any number of squares in any direction\*\*\*\*/

//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;

}

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;

}

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;

//in case of surrender, end the game first itself instead of checking validity

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

{

end\_game(old);

}

if(check\_king\_X(old,new)==0)

{

//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");

}

}

/\*\*\*\*following rules below must be applied irrespective of whether token is king or not\*\*\*\*/

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

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

{

check\_flag\_X=0;

help(old);

}

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, end the game first itself instead of checking validity

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

{

end\_game(old);

}

if(check\_king\_O(old,new)==0)

{

//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

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");

}

}

/\*\*\*\*following rules below must be applied irrespective of whether token is king or not\*\*\*\*/

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

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

{

check\_flag\_O=0;

help(old);

}

return check\_flag\_O;

}

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

\* End of check\_valid\_O() 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/bonus 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)//otherwise program won't end even after end\_game() function is activated

break;//we are using break in get\_turn() function since that's the last function to be executed in main() function

//so when break is applied here, main() function automatically returns 0 and program ends

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);

//activates to give 2nd chance if move made in original chance is 8 8

if(help(old))

{

help\_flag=0;//resetting to zero

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);

}

//activates to give 2nd chance if move made in original chance is invalid

if(check\_valid\_O(old,new))//in case move made is invalid, player gets 2nd chance

{

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 2nd Chance 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(king\_flag\_O==1)//bonus chance for player 2 to use king

{

move\_O(old, new);//to move token in original chance first

purple();

printf("\n %s's Turn (O's Bonus 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);

king\_flag\_O=0;//resetting to 0 since it should be active for only bonus turn

}

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);

//activates to give 2nd chance if move made in original chance is 8 8

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);

}

//activates to give 2nd chance if move made in original chance is invalid

if(check\_valid\_X(old,new))//in case move made is invalid, player gets 2nd chance

{

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 2nd Chance 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(king\_flag\_X==1)//bonus chance for player 1 to use king

{

move\_X(old, new);//to move token in original chance first

purple();

printf("\n %s's Turn (X's Bonus 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);

king\_flag\_X=0;//resetting to 0 since it should be active for only bonus turn

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");

green();

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

red();

printf("and");

green();

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

red();

printf("let's start the game. All the best!!!\*\*\*\n");

printf("\n");

blue();

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

printf("\n# Cutting of opponent's token: A player may \"push\" 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 player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.");

printf("\n He can move his token in all directions and any number of squares during bonus turn.\n But if he gets king in 2nd chance turn, bonus turn won't be given.");

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 #21

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

\* 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 \*

\* Notepad name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.txt \*

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

#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; //if king\_flag\_X=1, all X tokens will become king for 1 bonus turn

int king\_flag\_O=0; //if king\_flag\_O=1, all O tokens will become king for 1 bonus turn

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 \*

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

//displays scores of previous players (both winners and losers) using files and pointers concept of C language

void display\_prev\_score(struct player player\_X, struct player player\_O)

{

FILE \*fptr=(fopen("D:\\scoreboard.txt","a"));//"a" means appending mode---> type of mode for opened file

if(fptr==NULL){

red();

printf("Error!");

exit(1);

}

fprintf(fptr,"\n%s %d", player\_O.name,player\_O.count);

fprintf(fptr,"\n%s %d", player\_X.name,player\_X.count);

fclose(fptr);

}

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

\* End of display\_prev\_score() 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 \*

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

//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 player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.");

printf("\n He can move his token in all directions and any number of squares during bonus turn.\n But if he gets king in 2nd chance turn, bonus turn won't be given.");

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");

}

return help\_flag;

}

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

\* End of help() function \*

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

/\*\*\*\*if token becomes king, it can move any number of squares in any direction\*\*\*\*/

//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;

}

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;

}

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;

//in case of surrender, end the game first itself instead of checking validity

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

{

end\_game(old);

}

if(check\_king\_X(old,new)==0)

{

//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");

}

}

/\*\*\*\*following rules below must be applied irrespective of whether token is king or not\*\*\*\*/

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

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

{

check\_flag\_X=0;

help(old);

}

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, end the game first itself instead of checking validity

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

{

end\_game(old);

}

if(check\_king\_O(old,new)==0)

{

//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

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");

}

}

/\*\*\*\*following rules below must be applied irrespective of whether token is king or not\*\*\*\*/

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

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

{

check\_flag\_O=0;

help(old);

}

return check\_flag\_O;

}

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

\* End of check\_valid\_O() 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/bonus 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)//otherwise program won't end even after end\_game() function is activated and get\_turn() function will go in loop

break;//we are using break in get\_turn() function since that's the 2nd last function to be executed in main() function

//so when break is applied here, main() function automatically returns 0 after executing display\_prev\_score() function and program ends

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);

//activates to give 2nd chance if move made in original chance is 8 8

if(help(old))

{

help\_flag=0;//resetting to zero

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);

}

//activates to give 2nd chance if move made in original chance is invalid

if(check\_valid\_O(old,new))//in case move made is invalid, player gets 2nd chance

{

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 2nd Chance 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(king\_flag\_O==1)//bonus chance for player 2 to use king

{

move\_O(old, new);//to move token in original chance first

purple();

printf("\n %s's Turn (O's Bonus 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);

king\_flag\_O=0;//resetting to 0 since it should be active for only bonus turn

}

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);

//activates to give 2nd chance if move made in original chance is 8 8

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);

}

//activates to give 2nd chance if move made in original chance is invalid

if(check\_valid\_X(old,new))//in case move made is invalid, player gets 2nd chance

{

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 2nd Chance 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(king\_flag\_X==1)//bonus chance for player 1 to use king

{

move\_X(old, new);//to move token in original chance first

purple();

printf("\n %s's Turn (X's Bonus 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);

king\_flag\_X=0;//resetting to 0 since it should be active for only bonus turn

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");

/\*\*\*\*\*reading file for display of previous players' scores\*\*\*\*\*/

FILE \*fp;

struct players

{

char NAME[25];

int SCORE;

}player1[100],k;

int i=0,j=0,co=0;

int nli = 0;

char filename[100]="D:\\scoreboard.txt";

char c=0;

fp = fopen(filename, "r");

if (fp == NULL)

{

red();

printf("Could not open file %s", filename);

exit(0);

}

for (c = getc(fp); c != EOF; c = getc(fp))

if (c == '\n')

nli = nli + 1;//nli is number of lines in file called filename

fclose(fp);

FILE \*scoreboardfile1 = fopen("D:\\scoreboard.txt", "r");

/\*READ SCORES FROM TEXTFILE\*/

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

{

fscanf(scoreboardfile1, "%s %d \n",player1[i].NAME, &player1[i].SCORE);

}

for(i=0;i<nli;i++)//arranges scores in ascending order

{

for(j=0;j<nli-1;j++)

{

if(player1[j].SCORE<player1[j+1].SCORE)

{

k=player1[j];

player1[j]=player1[j+1];

player1[j+1]=k;

}

}

}

fclose(scoreboardfile1);

printf("\nPrevious Players' Score\n");

printf("\nRANK\t\t\tNAME\t\t\tSCORE\n");

printf("------------------------------------------------------------\n");

printf("TOP 3 PLAYERS \n");

for(i=0;i<10;i++)

{

if(i==3)

{

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

printf("%d\t\t\t%s\t\t\t%d\n",i+1,player1[i].NAME,player1[i].SCORE);

}

else

{

printf("%d\t\t\t%s\t\t\t%d\n",i+1,player1[i].NAME,player1[i].SCORE);

}

}

/\*\*\*\*\*End of reading file for display of previous players' scores\*\*\*\*\*/

printf("\n\n\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");

green();

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

red();

printf("and");

green();

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

red();

printf("let's start the game. All the best!!!\*\*\*\n");

printf("\n");

blue();

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

printf("\n# Cutting of opponent's token: A player may \"push\" 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 player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.");

printf("\n He can move his token in all directions and any number of squares during bonus turn.\n But if he gets king in 2nd chance turn, bonus turn won't be given.");

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);

display\_prev\_score(player\_X, player\_O);//calling this function to get latest players scores

return 0;

}

Progress #Final (took 40 days/5 weeks approx. to complete🡪 started on December 10 and ended on January 17)

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

\* 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 \*

\* Notepad name- Draught ASSIGNMENT1 #Samuela,Merlin,Deepiga,Sakthi.txt \*

\* Notepad used for storing scores of previous players- scoreboard.txt \*

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

#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; //if king\_flag\_X=1, all X tokens will become king for 1 bonus turn

int king\_flag\_O=0; //if king\_flag\_O=1, all O tokens will become king for 1 bonus turn

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 \*

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

//stores scores of previous players (both winners and losers) using files and pointers concept of C language

void store\_prev\_score(struct player player\_X, struct player player\_O)

{

FILE \*fptr=(fopen("D:\\scoreboard.txt","a"));//"a" means appending mode---> type of mode for opened file

if(fptr==NULL){

red();

printf("Error!");

exit(1);//exit(1) means program exited abnormally

}

//prints scores and names of player 2 and player 1 inside file

fprintf(fptr,"\n%s %d", player\_O.name,player\_O.count);

fprintf(fptr,"\n%s %d", player\_X.name,player\_X.count);

fclose(fptr);

}

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

\* End of store\_prev\_score() 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)//if current player who pressed 9 9 is player 1, player 2 will be declared as winner

{

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 \*

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

//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 player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.");

printf("\n He can move any of his tokens in all directions and any number of squares during bonus turn.\n But if he gets king in 2nd chance turn, bonus turn won't be given.");

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

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");

}

return help\_flag;

}

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

\* End of help() function \*

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

/\*\*\*\*if token becomes king, it can move any number of squares in any direction\*\*\*\*/

//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;

}

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;

}

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;

//in case of surrender, end the game first itself instead of checking validity

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

{

end\_game(old);

}

if(check\_king\_X(old,new)==0)

{

//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");

}

}

/\*\*\*\*following rules below must be applied irrespective of whether token is king or not\*\*\*\*/

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

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

{

check\_flag\_X=0;

help(old);

}

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, end the game first itself instead of checking validity

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

{

end\_game(old);

}

if(check\_king\_O(old,new)==0)

{

//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

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");

}

}

/\*\*\*\*following rules below must be applied irrespective of whether token is king or not\*\*\*\*/

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

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

{

check\_flag\_O=0;

help(old);

}

return check\_flag\_O;

}

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

\* End of check\_valid\_O() 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/bonus 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)//otherwise program won't end even after end\_game() function is activated and get\_turn() function will go in loop

break;//we are using break in get\_turn() function since that's the function which needs to stop executing as other functions are either directly or indirectly executed by calling inside this function only

//so when break is applied here, main() function automatically returns 0 after executing display\_prev\_score() function and program ends

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);

//activates to give 2nd chance if move made in original chance is 8 8

if(help(old))

{

help\_flag=0;//resetting to zero

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");

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);

}

//activates to give 2nd chance if move made in original chance is invalid

if(check\_valid\_O(old,new))//in case move made is invalid, player gets 2nd chance

{

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

blue();

printf("\n %s's Turn (O's 2nd Chance Turn)",player\_O.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\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(king\_flag\_O==1)//bonus chance for player 2 to use king

{

move\_O(old, new);//to move token in original chance first

yellow();

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

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

king\_flag\_O=0;//resetting to 0 since it should be active for only bonus turn

check\_flag\_O=0;//resetting to 0 since it should be active for only 2nd chance turn, otherwise it's showing as invalid even if player of next turn does valid move

}

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);

//activates to give 2nd chance if move made in original chance is 8 8

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");

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);

}

//activates to give 2nd chance if move made in original chance is invalid

if(check\_valid\_X(old,new))//in case move made is invalid, player gets 2nd chance

{

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

blue();

printf("\n %s's Turn (X's 2nd Chance Turn)",player\_X.name);

printf("\n Enter 9 9 as token row number and column number to surrender.\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(king\_flag\_X==1)//bonus chance for player 1 to use king

{

move\_X(old, new);//to move token in original chance first

yellow();

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

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

king\_flag\_X=0;//resetting to 0 since it should be active for only bonus turn

check\_flag\_X=0;//resetting to 0 since it should be active for only 2nd chance turn, otherwise it's showing as invalid even if player of next turn does valid move

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");

/\*\*\*\*\*reading file for display of previous players' scores\*\*\*\*\*/

FILE \*fp;

struct players

{

char NAME[25];

int SCORE;

}player1[100],k;

int i=0,j=0;

int nli = 0;

char filename[100]="D:\\scoreboard.txt";

char c=0;

fp = fopen(filename, "r");

if (fp == NULL)

{

red();

printf("Could not open file %s", filename);

exit(0);//exit(0) means program exited normally

}

for (c = getc(fp); c != EOF; c = getc(fp))//EOF means end of file, and getc is used to get characters in file

if (c == '\n')

nli = nli + 1;//nli is number of lines in file called filename

fclose(fp);

FILE \*scoreboardfile1 = fopen("D:\\scoreboard.txt", "r");

/\*READ SCORES FROM TEXTFILE LINE BY LINE\*/

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

{

fscanf(scoreboardfile1, "%s %d \n",player1[i].NAME, &player1[i].SCORE);

}

for(i=0;i<nli;i++)//arranges scores in descending order to display accordingly

{

for(j=0;j<nli-1;j++)

{

if(player1[j].SCORE<player1[j+1].SCORE)

{

k=player1[j];

player1[j]=player1[j+1];

player1[j+1]=k;

}

}

}

fclose(scoreboardfile1);

//displaying scores of previous players

printf("\nPrevious Players' Score\n");

printf("\nRANK\t\t\tNAME\t\t\tSCORE\n");

printf("------------------------------------------------------------\n");

printf("TOP 3 PLAYERS \n");

for(i=0;i<10;i++)

{

if(i==3)

{

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

printf("%d\t\t\t%s\t\t\t%d\n",i+1,player1[i].NAME,player1[i].SCORE);

}

else

{

printf("%d\t\t\t%s\t\t\t%d\n",i+1,player1[i].NAME,player1[i].SCORE);

}

}

/\*\*\*\*\*End of reading file for display of previous players' scores\*\*\*\*\*/

yellow();

printf("\n\n\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");

green();

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

red();

printf("and");

green();

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

red();

printf("let's start the game. All the best!!!\*\*\*\n");

printf("\n");

blue();

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

printf("\n# Cutting of opponent's token: A player may \"push\" 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 player's token reaches other end of the board, all of his tokens will become king and he'll be given a bonus turn.");

printf("\n He can move any of his tokens in all directions and any number of squares during bonus turn.\n But if he gets king in 2nd chance turn, bonus turn won't be given.");

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

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);//to start the game from turn 1

store\_prev\_score(player\_X, player\_O);//calling this function to get latest players scores and store in file

return 0;

}