CSE 102 – Computer Programming

Spring 2016

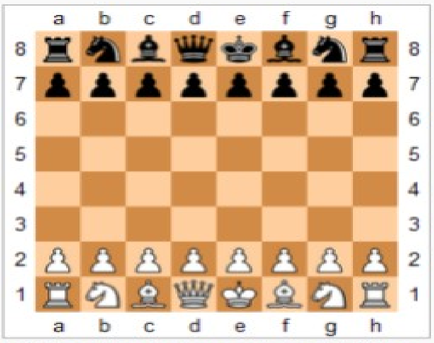
Homework #5

Due Date: April 6, 23:55

In this homework you will write a program to exercise with pointers, arrays and functions. Namely, you will write functions that performs basic operations of a chess simulator which will take commands and arguments from the user, executes the commands and return the result. This homework is an early stage and a simplified version of the game and will be completed by one of the future homeworks. Therefore it is crucial for you to have completed this homework by due date. Submit your homework via KADI.

Problem

You will write a chess simulator that performs basic operations in a game. We have an 8x8 board of which the columns and rows are represented by characters and integers, respectively as shown below.



Initial position: first row: rook, knight, bishop, queen, king, bishop, knight, and rook; second row: pawns

You will use lowercase and uppercase letters for the black and white users, respectively. You will display the board in the following form:

```
  a b c d e f g h
  - - - - - - - -
8|r n b q k b n r|
7|p p p p p p p p|
6|               |
5|               |
4|               |
3|               |
2|P P P P P P P P|
1|R N B Q K B N R|
  - - - - - - - -
```

You will store the board in a character array of size 64 and perform all of the operations using pointers.

The three main operations that can be performed in this stage of the game are 1) Initializing the board by placing the pieces in the correct order for both users, 2) Checking whether the piece can move from its position to a specified cell 3) Displaying the board 4) Exit.

A sample run of the program is given you as an attachment in the file “example_run.txt”. Note that in the example run, the last two “movable” commands (command number:2) fails because of the invalid inputs. The program will accept command from the user until he want to exit by entering command number 4.

To execute these commands you will need several helper functions as given below.

void getPosition(char *col, int *row) : This function will take the row and column values of a specific cell on the board.

int isValidCell(char col, int row) : This function will check whether the given cell is valid according to the board sizes and returns 1 if it is valid, otherwise 0.

void printBoard(char *board) : This function will print the current state of the board as show above.

int isPieceMovable(char *board) : It asks the user the source and target positions and check whether the piece in the source cell can be movable to the target cell. If the piece is movable it returns 1, otherwise 0. There are two things to consider while deciding a move is valida or not: 1) Whether the move is valid according to the general rules of the game, i.e. if there is another piece of the same player on the target cell, the move is not valid. 2) Whether the move is valid dependsing on the type of the piece in the source cell and the state of the board, e.g. a rock can not move to a cell if there is another piece on its path. Please note that, in this homework, ***<u>you will not move any piece but only check the validness of the move</u>***.

int isKnightMovable(char *board, char sourceCol, int sourceRow, char targetCol, int targetRow): This function will check whether a knight can move from source cell to the target cell according to the rules of the game.

int isRookMovable(char *board , char sourceCol, int sourceRow, char targetCol, int targetRow): This function will check whether a rook can move from source cell to the target cell.

…

The functions for other type of the pieces are similar and you will write each of them. For simplicity, assume that pawns always move one square straight forward.

The main function is provided you below to guide you writing your functions. The emptyBuffer() function is necessary to get rid of remnants of a command that could not be executed. For example, *<u>before initializing the board</u>*, if the user wants to execute the following command

>2 a5 b7

to check the movableness of the piece, the command cannot be executed. The arguments a5 and b7 become useless and should be read to empty the input buffer.

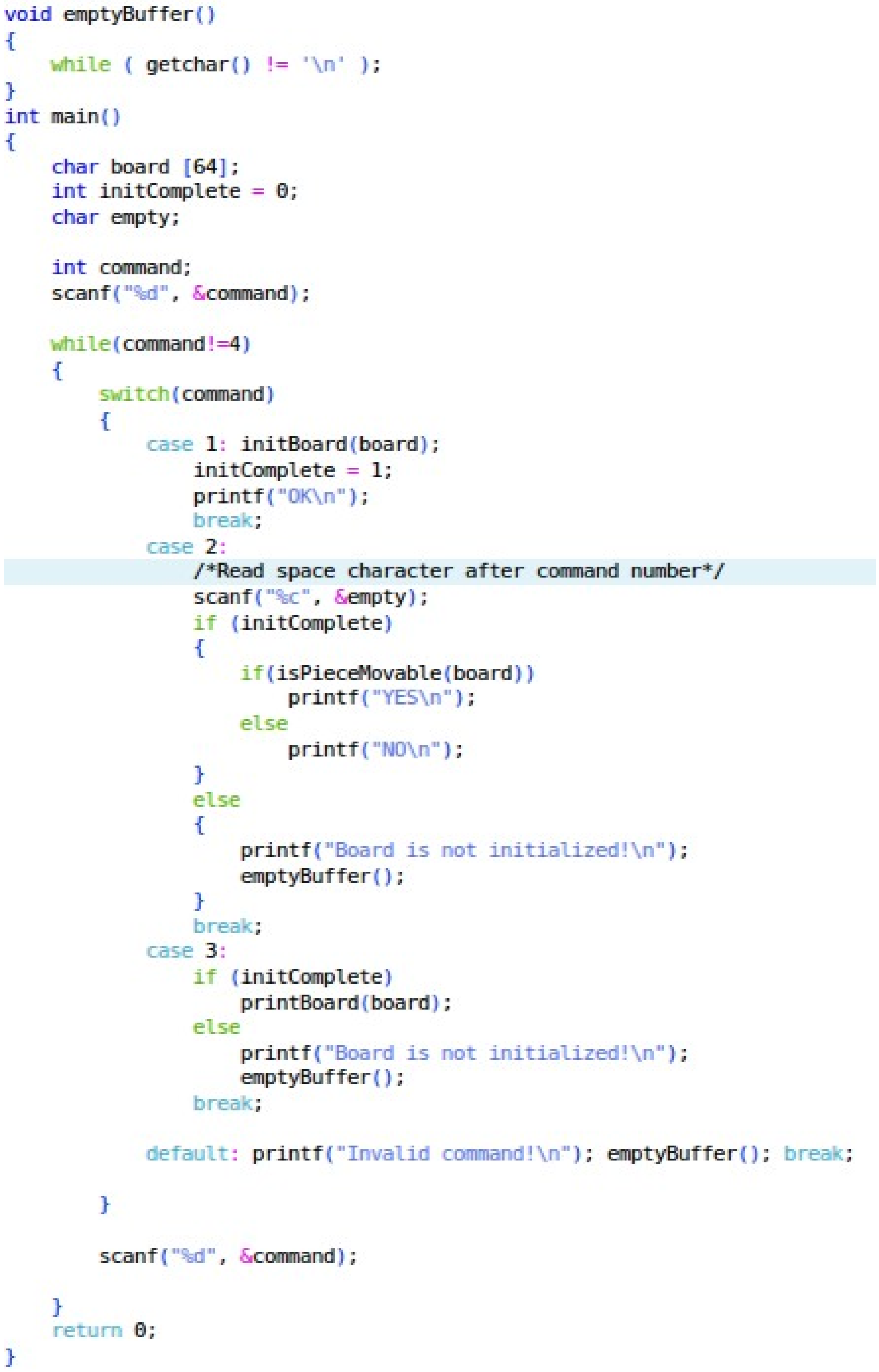
NOTES:

* Use the structures you learned such as loops, conditions and functions whenever needed. The

usage of these structures is as important as the correct execution of your program.

* The assignment must be your original work. Duplicate or very similar assignments are both going to be considered as cheating.

* Ask your questions via moodle.



```c
void emptyBuffer()
{
    while ( getchar() != '\n' );
}
int main()
{
    char board [64];
    int initComplete = 0;
    char empty;

    int command;
    scanf("%d", &command);

    while(command!=4)
    {
        switch(command)
        {
            case 1: initBoard(board);
                initComplete = 1;
                printf("OK\n");
                break;
            case 2:
                /*Read space character after command number*/
                scanf("%c", &empty);
                if (initComplete)
                {
                    if(isPieceMovable(board))
                        printf("YES\n");
                    else
                        printf("NO\n");
                }
                else
                {
                    printf("Board is not initialized!\n");
                    emptyBuffer();
                }
                break;
            case 3:
                if (initComplete)
                    printBoard(board);
                else
                    printf("Board is not initialized!\n");
                    emptyBuffer();
                break;

            default: printf("Invalid command!\n"); emptyBuffer(); break;

        }

        scanf("%d", &command);

    }
    return 0;
}
```