

Computer Organization

And

Assembly Language Lab

Project Report

2048 Game

**Group members**

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll no.** | **Section** |
| Abdul Wahab | 18F-0121 | C |
| Rizwan Ali | 18F-0106 | C |
| Haleema Sadia | 18F-0330 | C |

**National University of Computer & Emerging Sciences –FAST (FSD)**

1. **Game Introduction :**

*2048* is a single-player sliding blocks puzzles game designed by Italian web developer Gabriele Cirulli. The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number *2048*. If 2048 tile is found then game is won else if no empty tile left and none of the tile has 2048 then game is lost, Game displays appropriate message. On new game “2” number tile is inserted at two random positions.

1. **Features of Game :**
2. **Restart the Game :**

This Feature Restart the Game and generate number 2 on any two random blocks of 4X4 Grid.

1. **Undo the Game :**

This Feature Undo the Last move of the player.

1. **Counting the number of moves :**

This Feature counts the number of player’s moves.

1. **Generating number 2 at random position :**

This Feature generates number 2 at random position in any empty block of 4x4 Grid on every move of the player only if tiles have changed position on movement.

1. **Check for Win or Lose :**

This Feature checks the block that contains 2048 in 4x4 Grid on every move of the player If found win else if blocks are full and 2048 is not found then game is lost.

1. **Use of Arrow keys / Escape key :**

This Feature moves the blocks according to the key (arrow keys) pressed by the player and Exit the game by pressing Esc key.

1. **File handling to maintain the number of moves :**

This Feature stores the high score in the form of number of moves .

1. **Resume the Game :**

This Feature Resume the game. This Feature re pick the data from file.

1. **Coloring :**

This Feature uses built in functions to set colors of each line and formatting the output with proper spacing on console

1. **Menu Driven:**

Game is menu driven with all basic functions enlisted. User can go back and forth between various functions also it avoid invalid input choice of user.

**CODE:**

Include Irvine32.inc

.data

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; GameLogo

log1 DB " | \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_ \_\_\_ \_\_\_\_\_\_ |",0

log2 DB " | / \_\_ \ / \_\_ \ | | | | / \_\_ \ |",0

log3 DB " | |\_| | | | | | | | |\_\_\_| | | |\_\_| | |",0

log4 DB " | \_\_\_\_| | | | | | |\_\_\_\_\_ | | \_\_ | |",0

log5 DB " | / \_\_\_\_/ | | | | | | | | | | |",0

log6 DB " | | |\_\_\_\_ | |\_\_| | | | | |\_\_| | |",0

log7 DB " | |\_\_\_\_\_\_| \\_\_\_\_\_\_/ |\_| \\_\_\_\_\_\_/ |",0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Loading

temp1 DD ?

temp2 DD ?

MyStr0 db " Loading ...",0

stmsg db " (Press any key to start the Game)",0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; BOARD

board word 16 dup(0)

tempBoard word 16 dup(0)

boardSize=$-tempBoard

filename1 db "highscore.txt",0

filename db "boardsave.txt",0

fileHandle DWORD ?

bytesWritten dword ?

temp word 4 dup(0) ; used in movement and removing zeroes

noMoves dw 0

ban1 db "================",0

ban2 db "===========================================",0

ban3 db " ",0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Gameplay instructions

inst1 db "Number of 'moves': ",0

inst2 db "Enter your Choice: ",0

inst3 db "Press 'Arrow Keys' for movement ",0

inst4 db "Press 'U' or 'u' to undo a move ",0

inst5 db "Press 'Esc' to Endgame ",0

inst6 db "HURRAYY!! 'YOU WON!!'",0

inst7 db "ALAS!! 'YOU LOST!!'",0

rowNo dd 0

colNo dd 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Game Title

GameName1 db " 2048 Game : ",0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Menu Title

GameName db " 2048 Game : ",0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Menu

MyStr db " Menu : ",0

MyString db "1. Play Game ",0

MyString0 db "2. Resume ",0

MyString1 db "3. How to play ",0

MyString2 db "4. High Score ",0

MyString3 db "5. Exit(Press '5' or 'Esc' for Exit) ",0

input db ?

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Instructions How to play

Stat1 db " 1) 2048 is an addictive puzzle game based on simple addition.",0dh,0ah,0

stat2 db " 2) The objective of the game is to reach 2048 by merging adjacent similar number tiles on a 4x4 board.",0dh,0ah,0

stat3 db " 3) The game starts with two tiles of 2 random positions on the board ",0

stat4 db " and continues to add new tiles of 2 on random tiles after every move.",0dh,0ah,0

stat5 db " 4) The game ends when your board is completely filled with numbers and you donot have any move left.",0dh,0ah

db " or you have summed up a tile to 2048. ",0dh,0ah,0dh,0ah

db " 5) Use arrow keys for movement and escape key to exit. ",0dh,0ah,0dh,0ah

db " Programmed by: ",0dh,0ah,0dh,0ah

db " 1. ABDUL WAHAB (18F-0121)",0dh,0ah

db " 2. RIZWAN ALI (18F-0106)",0dh,0ah

db " 3. HALEEMA SADIA (18F-0330)",0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

.code

main proc

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; LOGO

Call Logo

mov edx,offset filename1 ;loading highscore at the start for resume

call OpenInputFile

mov fileHandle,eax

mov edx,offset noMoves

mov ecx,2 ; reading two bytes from file

call ReadFromFile

jc notFound

mov eax,fileHandle ;closing file in reading mode

call closeFile

mov edx,offset filename ; loading board with values to use resume function

call OpenInputFile

mov fileHandle,eax

mov edx,offset board

mov ecx,boardSize

call ReadFromFile

jc notFound

mov eax,fileHandle

call closeFile

mov esi,offset board

mov edi,offset tempboard

mov ecx,16

cpy:

lodsw

stosw

loop cpy

jmp next1 ;resume file found

notFound: ;no resume txt file available call newgame

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; New game

call newgame

next1:

call Menu

call ReadChar ; read choice

Exit

main endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;logo print func

Logo proc

MOV EAX,0

MOV AL,01h

Call setTextColor ;Set Black bg and Blue text (Logo)

MOV EDX,OFFSET log1

Call WriteString

Call Crlf

MOV EDX,OFFSET log2

Call WriteString

Call Crlf

MOV EDX,OFFSET log3

Call WriteString

Call Crlf

MOV EDX,OFFSET log4

Call WriteString

Call Crlf

MOV EDX,OFFSET log5

Call WriteString

Call Crlf

MOV EDX,OFFSET log6

Call WriteString

Call Crlf

MOV EDX,OFFSET log7

Call WriteString

Call Crlf

Call Loading

ret

Logo endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Loading proc

MOV EAX,0

MOV AL,05h

Call setTextColor ;Set Black bg and Purple text

MOV EDX,OFFSET ban3

Call WriteString

MOV EDX,OFFSET ban2

Call WriteString

Call Crlf

MOV EAX,0

MOV AL,04h

Call setTextColor ;Set Black bg and Red text

MOV EDX,OFFSET ban3

Call WriteString

MOV EDX,OFFSET Mystr0

Call WriteString

Call Crlf

MOV EAX,0

MOV AL,05h

Call setTextColor ;Set Black bg and Purple text

MOV EDX,OFFSET ban3

Call WriteString

MOV EDX,OFFSET ban2

Call WriteString

Call Crlf

MOV EAX,0

MOV AL,04h

Call setTextColor ;Set Black bg and Red text

MOV EDX,OFFSET ban3

Call WriteString

Call star

Call Crlf

MOV EAX,0

MOV AL,05h

Call setTextColor ;Set Black bg and Purple text

MOV EDX,OFFSET ban3

Call WriteString

MOV EDX,OFFSET ban2

Call WriteString

Call Crlf

MOV EAX,0

MOV AL,04h

Call setTextColor ;Set Black bg and Red text

MOV EDX,OFFSET ban3

Call WriteString

MOV EDX,OFFSET stmsg

Call WriteString

Call Crlf

Call ReadChar

ret

Loading endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Star proc

MOV ECX,43

l1:

MOV temp1,ECX

MOV ECX,0000FFFEh

l2:

MOV temp2,ECX

MOV ECX,0001FFh

l3:

;Delay

loop l3

MOV ECX,temp2

loop l2

MOV ECX,temp1

MOV AL,'\*' ; Loading \*\*\*

Call WriteChar

loop l1

ret

Star endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Gametitle proc

MOV EAX,0

MOV AL,0B5h

Call setTextColor ;Set Light Sky Blue bg and Purple text

call writeHBoarder

call writeHBoarder

call writeHBoarder

call writeHBoarder

call S\_space

call crlf

MOV EAX,0

MOV AL,0B4h

Call setTextColor ;Set Light Sky Blue bg and Red text

mov edx,offset GameName1

call writeString

call crlf

MOV EAX,0

MOV AL,0B5h

Call setTextColor ;Set Light Sky Blue bg and Purple text

call writeHBoarder

call writeHBoarder

call writeHBoarder

call writeHBoarder

call S\_space

call crlf

ret

Gametitle endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Gametitle2 proc

mov edx,offset ban3

call writeString

MOV EAX,0

MOV AL,05h

Call setTextColor

call writeHBoarder

call crlf

mov edx,offset ban3

call writeString

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx,offset GameName

call writeString

call crlf

mov edx,offset ban3

call writeString

MOV EAX,0

MOV AL,05h

Call setTextColor

call writeHBoarder

call crlf

ret

Gametitle2 endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Menu PROC

E:

call clrscr

call Gametitle2

MOV EAX,0

MOV AL,06h

Call setTextColor

call menuformat

mov edx,offset Mystr

call writestring

call menuformat

mov edx, offset MyString

call writestring

call menuformat

mov edx, offset MyString0

call writestring

call menuformat

mov edx, offset MyString1

call writestring

call menuformat

mov edx, offset MyString2

call writestring

call menuformat

mov edx, offset MyString3

call writestring

call menuformat

mov edx, offset inst2

call writestring

call crlf

mov edx, offset ban3

call writestring

MOV EAX,0

MOV AL,05h

Call setTextColor

mov edx, offset ban2 ;lower upper borders

call writestring

Call Crlf

mov edx, offset ban3

call writestring

call readchar

mov input,al

cmp al,'1'

je l1

jne t0

l1: ;to call Play Function

call Play

jmp next

t0:

cmp input,'2'

je l0

jne t1

l0: ;to call resume Function

call resume

jmp next

t1:

cmp input,'3'

je l2

jne t2

l2: ;to call HOW\_TO\_PLAY Function

call HOW\_TO\_PLAY

jmp next

t2:

cmp input,'4'

je l3 ;to call High\_Score Function

jne t3

l3:

call High\_Score

jmp next

t3:

cmp input,'5'

je Ex ;to call Exit Function

jne t4

t4:

mov AL,input

cmp AX,011Bh

je Ex ;to call Exit Function

jne E

EX:

call Ext

next:

jMP E

ret

Menu endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

menuformat proc

call crlf

mov edx,offset ban3

call writestring

MOV EAX,0

MOV AL,05h

Call setTextColor

mov edx,offset ban2

call writestring

call crlf

MOV EAX,0

MOV AL,06h

Call setTextColor

mov edx,offset ban3

call writestring

call writeVBoarder

ret

menuformat endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Play PROC

call clrscr

call newGame ;initializes board and moves with new game values

call performOper ; main function that reads char and performs operations like left right etc

ret

Play endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

resume PROC ;//////

call clrscr ; resume values are loaded

call performOper

ret

resume endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

HOW\_TO\_PLAY PROC

call clrscr

MOV EAX,0

MOV AL,05h

Call setTextColor

call writeHBoarder

call crlf

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx,offset MyString1

call writeString

call crlf

MOV EAX,0

MOV AL,05h

Call setTextColor

call writeHBoarder

call crlf

mov edx,offset Stat1

call writeString

call CRLF

mov edx,offset Stat2

call writeString

call CRLF

mov edx,offset Stat3

call writeString

call CRLF

mov edx,offset Stat4

call writeString

call CRLF

mov edx,offset Stat5

call writeString

call CRLF

Call readChar

ret

HOW\_TO\_PLAY endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

High\_Score PROC

call clrscr

MOV EAX,0

MOV AL,05h

Call setTextColor

call writeHBoarder

call crlf

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx,offset MyString2

call writeString

mov al,':'

call writeChar

movzx eax, noMoves

call writeDec

call crlf

MOV EAX,0

MOV AL,05h

Call setTextColor

call writeHBoarder

call crlf

Call readChar

ret

High\_Score endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Ext proc

mov edx, offset filename

call createOutputFile

mov fileHandle,eax

mov edx,offset Board

mov ecx,boardSize

call writeToFile

mov edx, offset filename1

call createOutputFile

mov fileHandle,eax

mov edx,offset noMoves

mov ecx,2

call writeToFile

Exit

Ext endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

newGame proc

mov eax,0

mov noMoves,0

mov ax,0

mov ecx,16

mov edi,offset board

rep stosw

call insert2

call insert2

mov esi, offset board

mov edi,offset tempBoard

mov ecx,16

l1:

lodsw

stosw

loop l1

call printboard

ret

newGame endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; gives random position where 2 should be inserted gives index as even number

insert2 proc

again:

mov eax, 32 ; max range of random position

call randomize

call randomRange ; 0 to 31

mov ebx,eax

clc

shr ebx,1 ;check if even

jnc next

inc eax

cmp eax,32

jb next

mov eax,0

next:

cmp board[eax],0

je next1

add eax,2

cmp eax,32

jb next2

mov eax,0

next2: jmp next

next1:

mov board[eax],2d

ret

insert2 endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

printBoard proc

call clrscr

call Gametitle

mov esi, offset board

mov ecx, 4

mL:

MOV EAX,0

MOV AL,0C5h

Call setTextColor

call writeHBoarder

call writeHBoarder

call writeHBoarder

call writeHBoarder

call S\_space

call crlf

push ecx

mov ecx,4

call writeVBoarder

call writespace

inL:

MOV EAX,0

MOV AL,0E0h

Call setTextColor

MOV EAX,0

mov ax, [esi]

call writeDec

MOV EAX,0

MOV AL,0C5h

Call setTextColor

add esi, type board

call writespace

call writeVBoarder

CMP CL,1

je SKP

call writespace

SKP:

loop inL

call crlf

pop ecx

loop mL

call writeHBoarder

call writeHBoarder

call writeHBoarder

call writeHBoarder

call S\_space

call crlf

call printInstr

ret

printBoard endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

S\_space proc

mov al, ' '

call writechar

ret

S\_space endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

writespace proc

mov al, TAB

call writechar

ret

writespace endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

writeHBoarder proc

mov edx,offset ban1

call writestring

ret

writeHBoarder endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

writeVBoarder proc

mov al, 7Ch

call writechar

ret

writeVBoarder endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Hline proc

call crlf

MOV EDX,offset ban2

Call writestring

call crlf

ret

Hline endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

format proc

MOV EAX,0

MOV AL,05h

Call setTextColor

Call Hline

call writeVBoarder

ret

format endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

printInstr proc

call format

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx, offset inst1

call writeString

movzx eax, noMoves

call writeDec

call format

MOV EAX,0

MOV AL,0Ah

Call setTextColor

mov edx, offset inst3

call writeString

call format

MOV EAX,0

MOV AL,0Eh

Call setTextColor

mov edx, offset inst4

call writeString

call format

MOV EAX,0

MOV AL,06h

Call setTextColor

mov edx, offset inst5

call writeString

call format

MOV EAX,0

MOV AL,03h

Call setTextColor

mov edx, offset inst2

call writeString

MOV EAX,0

MOV AL,05h

Call setTextColor

Call Hline

MOV EAX,0

MOV AL,05h

Call setTextColor

ret

printInstr endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

performOper proc

again:

call printBoard

mov eax,0

call readChar

cmp ax,4800h ;up arrow

je arrow\_up

cmp ax,5000h ;down arrow

je arrow\_Down

cmp ax,4D00h ;right arrow

je arrow\_Right

cmp ax,4B00h ;left arrow

je arrow\_left

cmp ax,011Bh ;escape key

je outt

cmp al,'U' ; undo

je undoo

cmp al,'u'

je undoo

jmp endd ; invalid key input

arrow\_up:

inc noMoves

call copyBoard ; copy board for undo operations for board to tempboard

call movUp ; all operations related to mov up

jmp endd

arrow\_Down:

inc noMoves

call copyBoard

call movDown

jmp endd

undoo:

call reCopyBoard ; copy board from temp board to board

jmp endd

arrow\_left:

inc noMoves

call copyBoard

call movLeft

jmp endd

arrow\_Right:

inc noMoves

call copyBoard

call movRight

endd:

call checkWin ; check for win then out of perform operation

jz outt

call checkEmpty ; check for checkempty if not empty means lost

jnz outt

call insertVals ;after every movement inserts2 at random position iff tiles have shifted there position

jmp again ;again repeat process

outt:

ret

performOper endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

copyBoard proc

mov esi , offset board

mov edi, offset tempboard

mov ecx,16

mL:

lodsw

stosw

loop mL

ret

copyBoard endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

movUp proc

mov colNo,0

mov ecx, 4

moveL:

call removeZerosU

call addUp

call removeZerosU

add colNo,2

loop moveL

ret

movUp endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

removeZerosU proc uses ecx

mov esi,colNo

mov ecx, 4

mov edi, 0

mL:

cmp board[esi],0

je next

mov bx, board[esi]

mov temp[edi],bx

add edi, type board

next:

add esi,8

loop mL

mov esi,colNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov board[esi],bx

mov temp[edi],0

add esi, 8

add edi, type board

loop copyagain

ret

removeZerosU endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

addUp proc uses ecx

mov esi, colNo

mov ecx,3

l1:

mov bx, board[esi]

cmp bx,board[esi+8]

jne next

shl bx,1

mov board[esi],bx

mov board[esi+8],0

next:

add esi,8

loop l1

ret

addUp endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

movDown proc

mov colNo,0

mov ecx, 4

moveD:

call removeZerosD

call addDown

call removeZerosD

add colNo,2

loop moveD

ret

movDown endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

removeZerosD proc uses ecx

mov esi,colNo

add esi,24

mov ecx, 4

mov edi, 6

mL:

cmp board[esi],0

je next

mov bx, board[esi]

mov temp[edi],bx

sub edi, type board

next:

sub esi,8

loop mL

mov esi,colNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov board[esi],bx

mov temp[edi],0 ;reinitializing temp to zero for further moves

add esi, 8

add edi, type board

loop copyagain

ret

removeZerosD endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

addDown proc uses ecx

mov esi, colNo

add esi,24

mov ecx,3

l1:

mov bx, board[esi]

cmp bx,board[esi-8]

jne next

shl bx,1

mov board[esi],bx

mov board[esi-8],0

next:

sub esi, 8

loop l1

ret

addDown endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

reCopyBoard proc

mov esi , offset tempboard

mov edi, offset board

mov ecx,16

mL:

lodsw

stosw

loop mL

ret

reCopyBoard endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

movLeft proc

mov rowNo,0

mov ecx, 4

moveL:

call removeZerosL

call addLeft

call removeZerosL

add rowNo,8

loop moveL

ret

movLeft endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

removeZerosL proc uses ecx

mov esi,rowNo

mov ecx, 4

mov edi, 0

mL:

cmp board[esi],0

je next

mov bx, board[esi]

mov temp[edi],bx

add edi, type board

next:

add esi, type board

loop mL

mov esi,rowNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov board[esi],bx

mov temp[edi],0

add esi, type board

add edi, type board

loop copyagain

ret

removeZerosL endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

addLeft proc uses ecx

mov esi, rowNo

mov ecx,3

l1:

mov bx, board[esi]

cmp bx,board[esi+2]

jne next

shl bx,1

mov board[esi],bx

mov board[esi+2],0

next:

add esi,type word

loop l1

ret

addLeft endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

movRight proc

mov rowNo,0

mov ecx, 4

moveR:

call removeZerosR

call addRight

call removeZerosR

add rowNo,8

loop moveR

ret

movRight endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

removeZerosR proc uses ecx

mov esi,rowNo

add esi,6

mov ecx, 4

mov edi, 6

mL:

cmp board[esi],0

je next

mov bx, board[esi]

mov temp[edi],bx

sub edi, type board

next:

sub esi, type board

loop mL

mov esi,rowNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov board[esi],bx

mov temp[edi],0

add esi, type board

add edi, type board

loop copyagain

ret

removeZerosR endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

addRight proc uses ecx

mov esi, rowNo

add esi,6

mov ecx,3

l1:

mov bx, board[esi]

cmp bx,board[esi-2]

jne next

shl bx,1

mov board[esi],bx

mov board[esi-2],0

next:

sub esi,type word

loop l1

ret

addRight endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

checkWin proc uses ecx

mov ecx,16

mov ax,2048

mov edi, offset board

repne scasw

jnz endd

call printBoard

call crlf

call crlf

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx, offset inst6

Call WriteString

Call ReadChar

endd:

ret

checkWin endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

checkEmpty proc uses ecx

mov ecx,16

mov ax,0

mov edi, offset board

repne scasw

jz endd

call printBoard

call crlf

call crlf

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx, offset inst7

Call WriteString

Call ReadChar

endd: ret

checkEmpty endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

insertVals proc

call notChanged

jz endd

call insert2

endd:

ret

insertVals endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

notChanged Proc

mov ecx,16

mov esi,offset Board

mov edi , offset tempBoard

repe cmpsw

ret

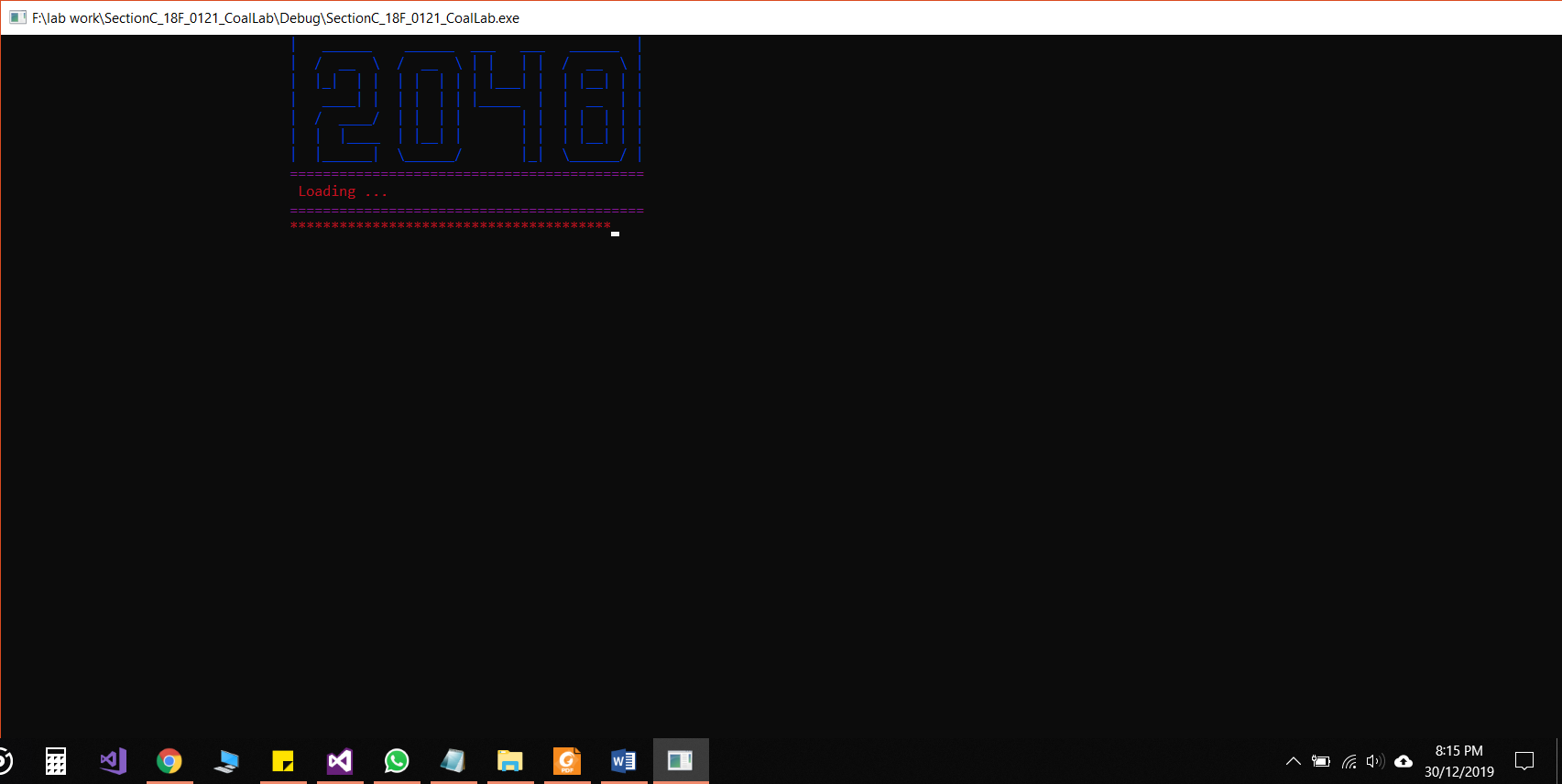
notChanged endp

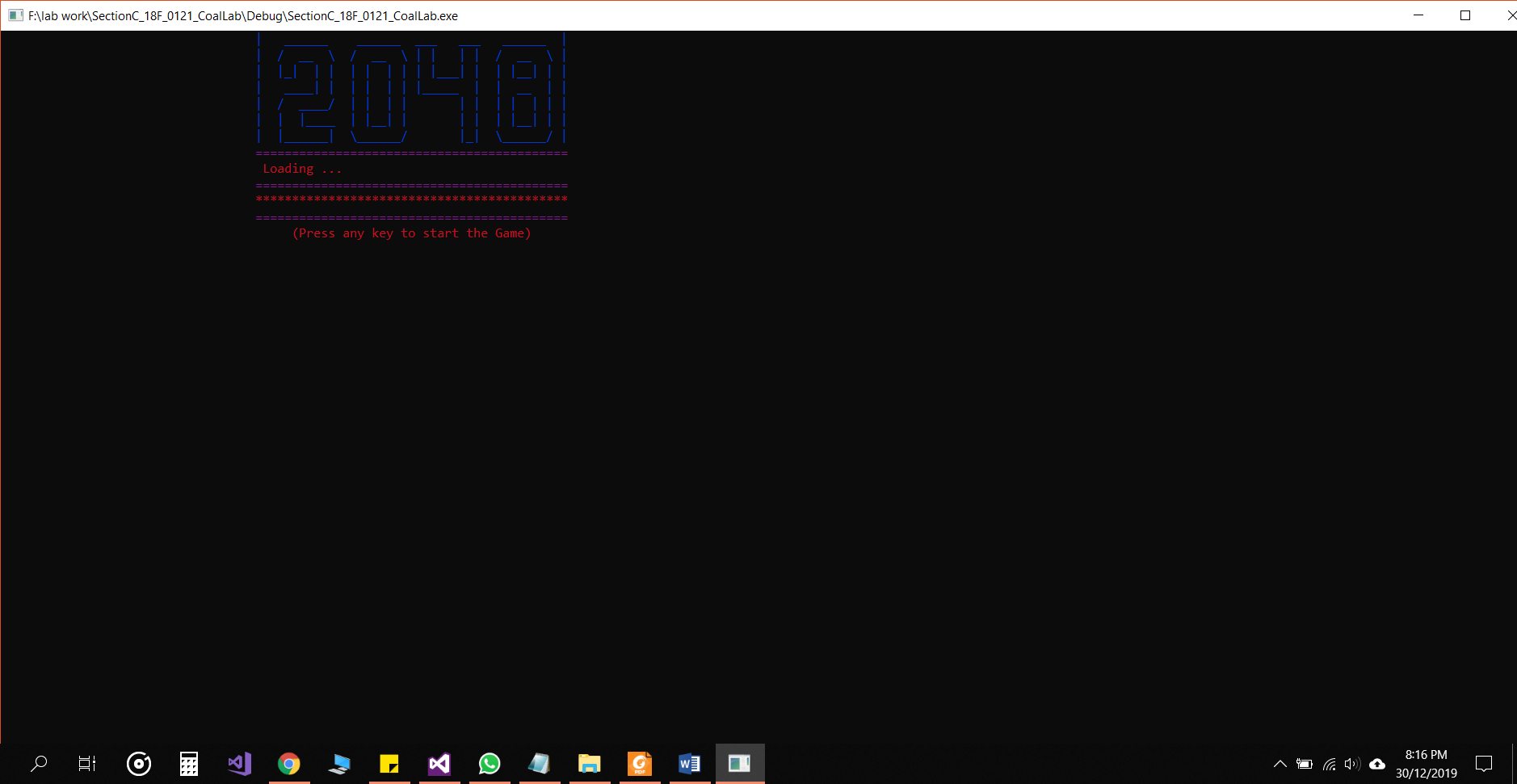
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

End main

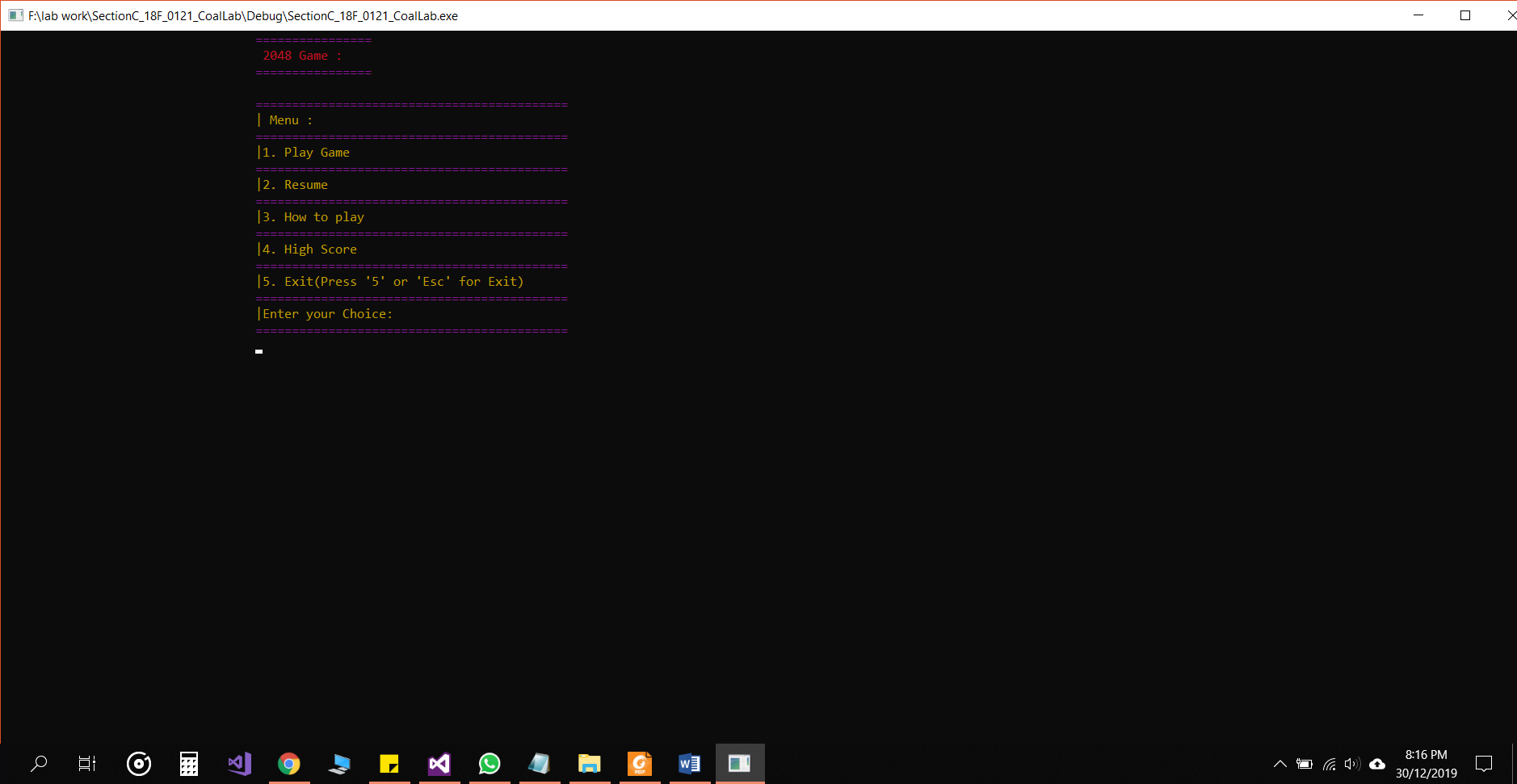
**GAMEPLAY:**

**Loading animation:**

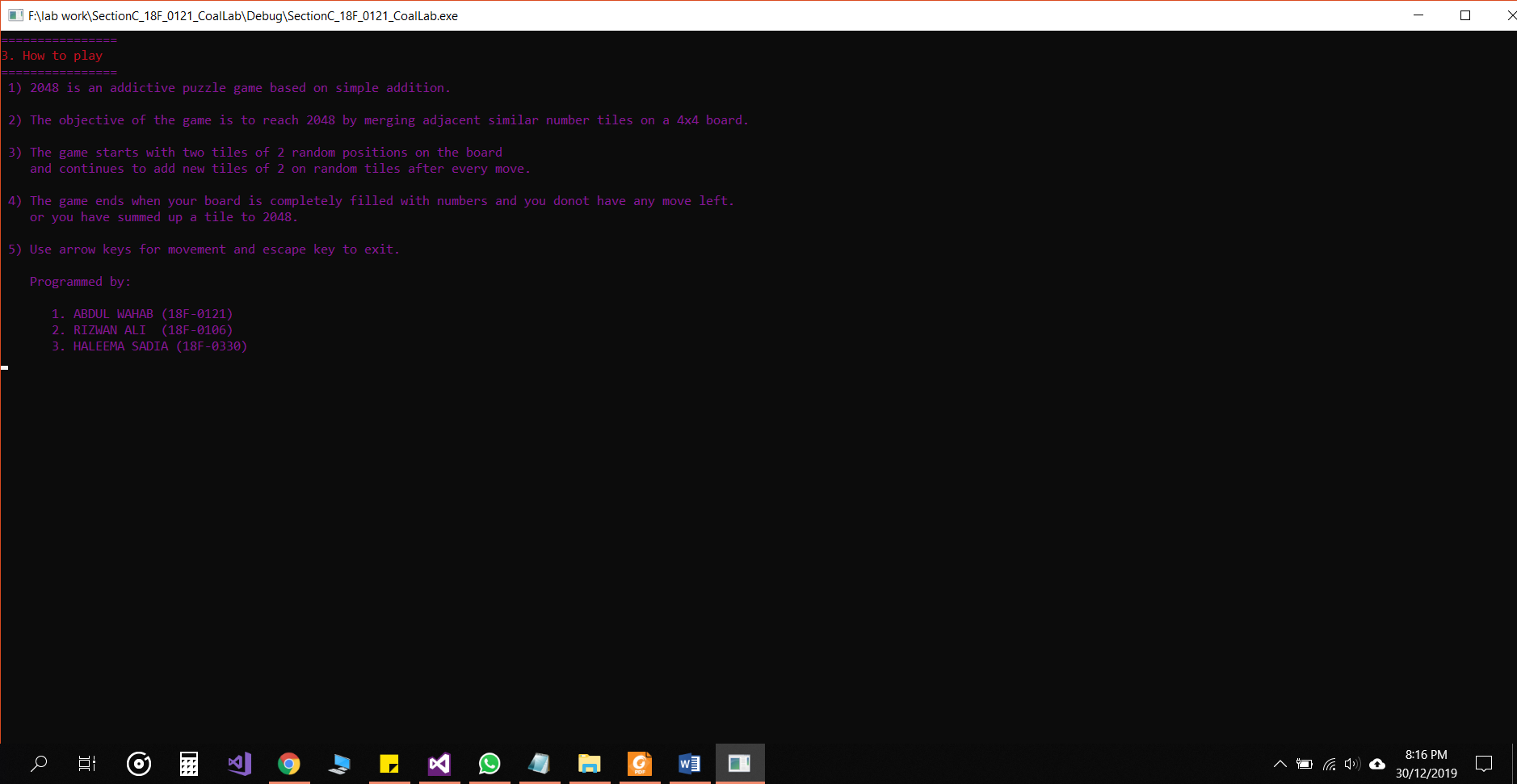


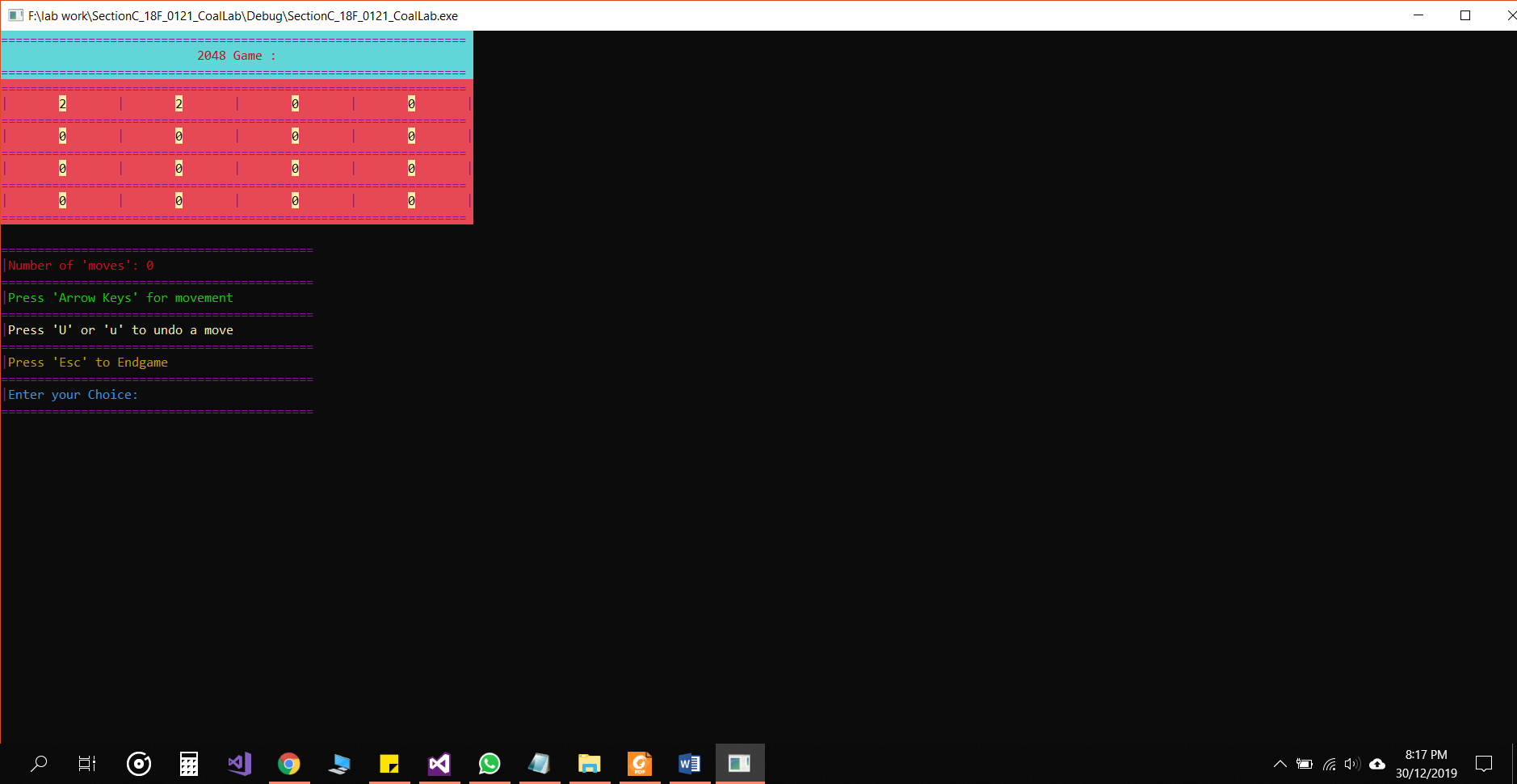


**Main menu:**

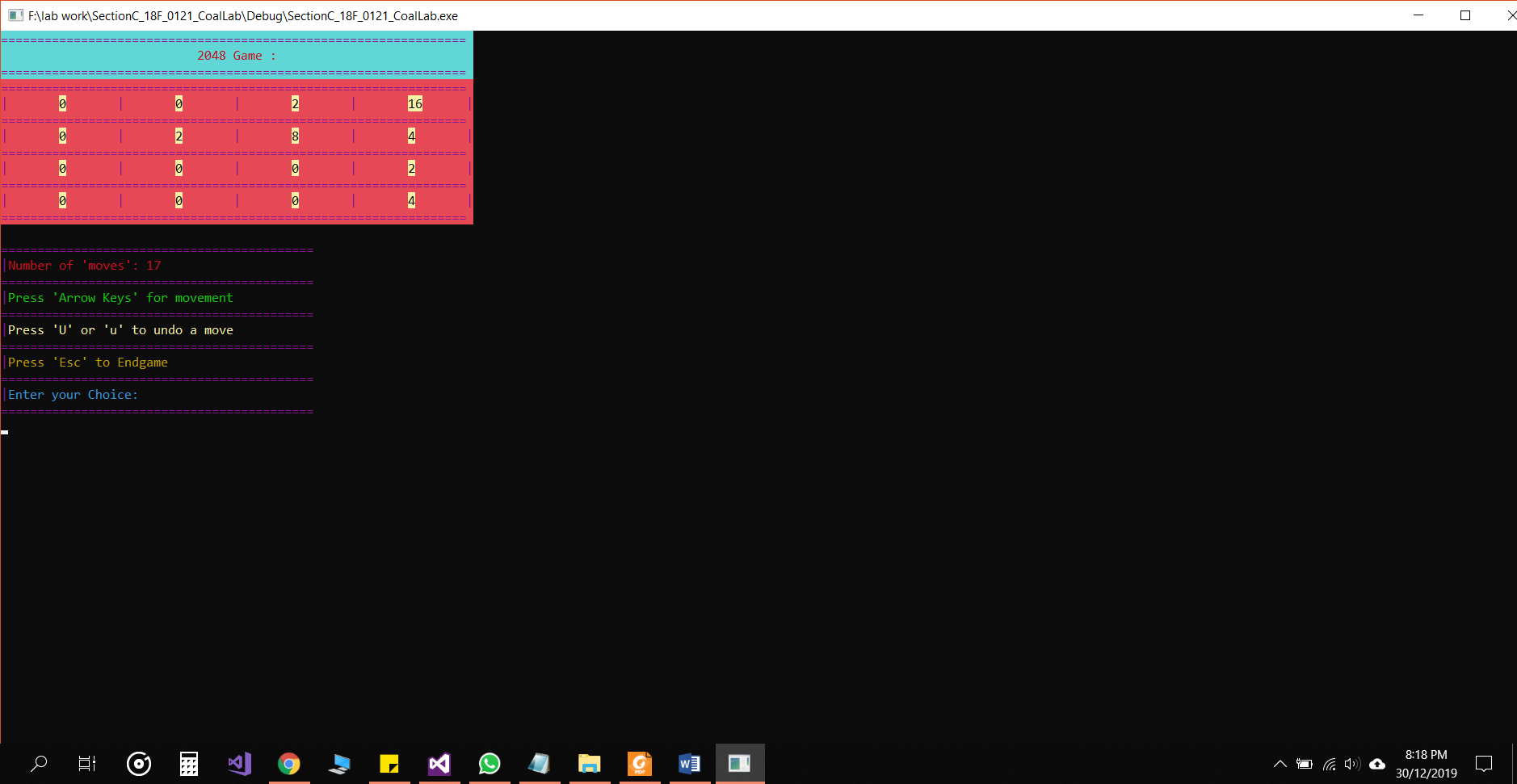


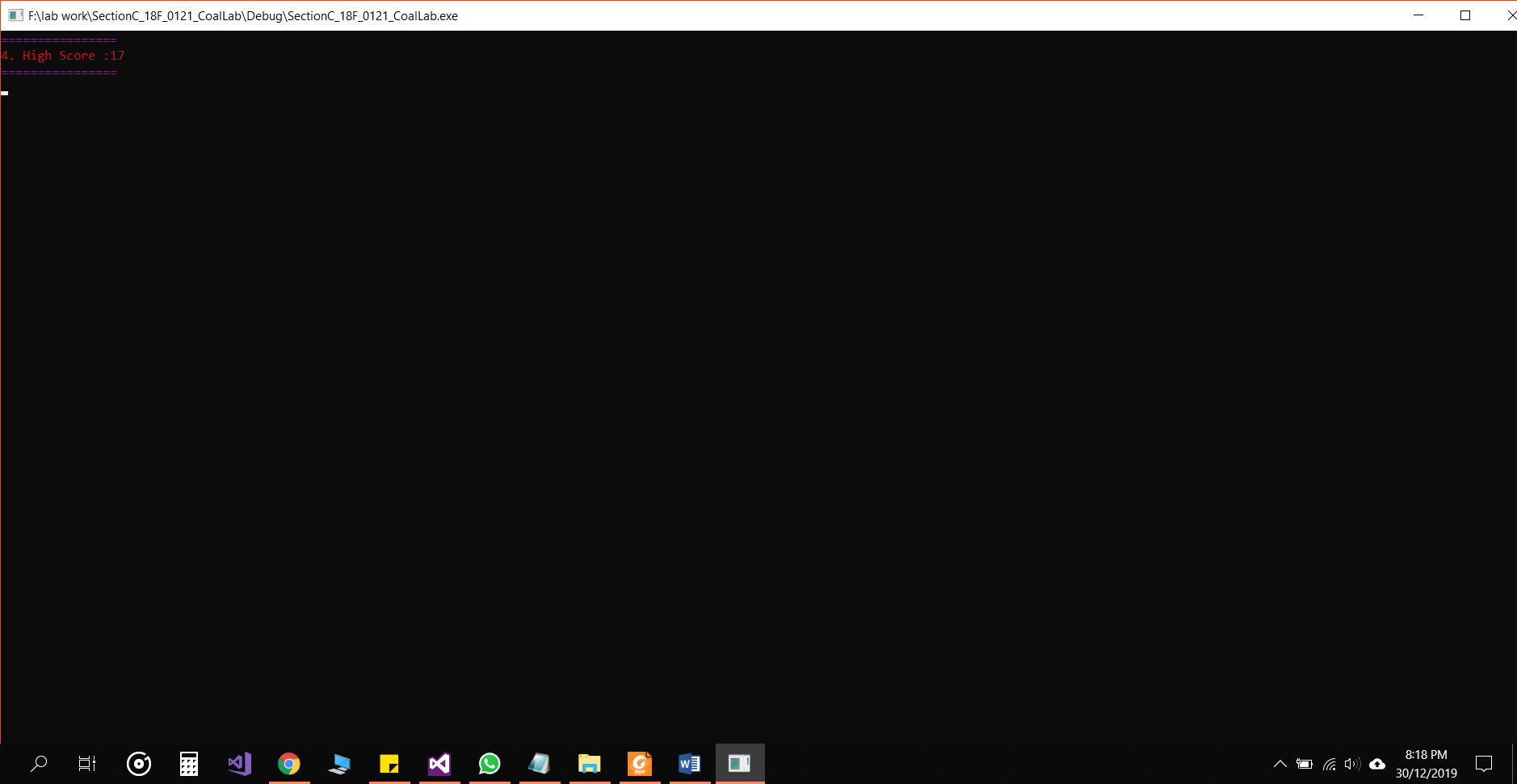
**How to play:**



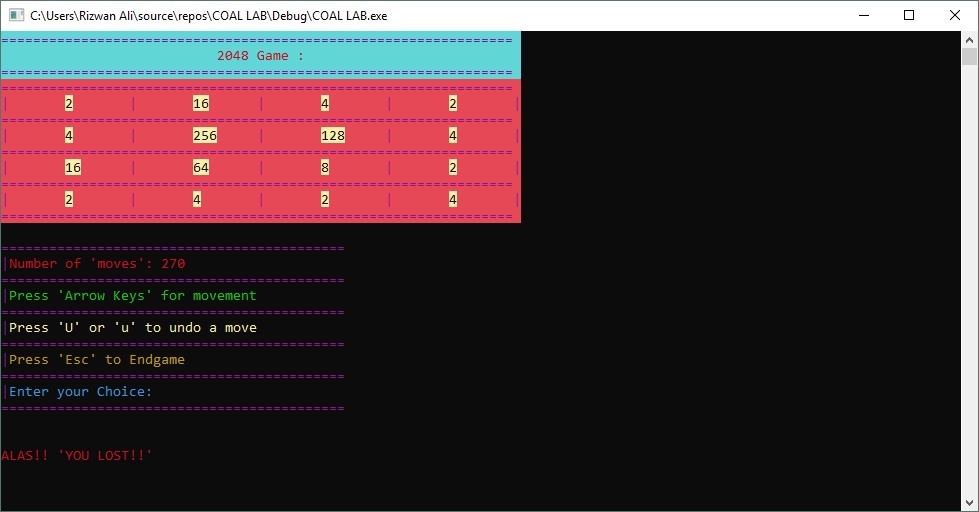


**Gameplay to previous condition after exiting and resuming:**





**Showing appropriate winning and losing message:**

****