**Project: Snake game**

****

**Group Members**

**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES - FAST**

The aim of this project is to increase our knowledge of fundamental concepts of probability and statistics.and utilize whaevert we have learned in classroom with real world applications

**Introduction**

The aim of this project was to create a Fun and simple traditional Snake game which have all the mechanics of a standard snake game which we used to play in our childhood and to refresh and implement all of our knowledge of assembly language .this game have 3 screens first the home screen then the game boundaries and at the the end a screen showing total points and game status .

The motivation behind choosing this specific project was to recall the hidden and forgotten benefits of this game as new games have taken all over childrens

* **This is not just a game it is a philosophy of life it teaches us how you could be using your own success against you when you become so long that you get in your own way**
* **Snake teaches us tenacity**
* **The ability to fail and learn from the setback and move forward is a virtual life skill.**
* **Learning that failure is not the end but part of the learning experience**

**Overview**

Snake first appeared in 1997 on the Nokia 6110, along with the games Logic and Memory. It was programmed by Taneli Armanto, a design engineer at Nokia. The concept originated from the 1976 arcade game ‘Blockade’, developed and published by Gremlin.

Pekka Rantala, Chief Marketing Officer of HMD Global said:

*“The Snake game was the most popular game in the history of Nokia phones, in fact the true origin of mobile gaming can be traced back to a Nokia handset released in 1997. To this day it has remained hugely popular, not only because it carries a strong sense of nostalgia but because this was the first ever mobile gaming experience for many consumers. The opportunity to partner with Facebook, the social media platform that has single handedly defined the social media landscape, is an incredible opportunity. We were truly excited to work with them to bring back Snake in a modern format to a global audience. We hope in its new format it unites people across the world and brings as much joy to players as it did the first time around.”*

**Problem Statement:**

The problem is to design a Snake Game which provides the following functionalities:

* Snake can move in any direction
* Food will be generated at a given interval.
* Its length grows when it eats food
* When snake hits the boundary the game is over
* When the snake crosses itself, the game will be over.
* Apart from them there were three major problems in this project that we have discussed in the implementation and testing section

.

The main procedures used are:

* Game loop
* Draw body
* Scoreboard
* GameBoundary
* Snake Speed
* Creating Coin

//The above mention names may vary in program but functionalities are same

**Methodology**

* ***Create the Screen.***
* ***Create the Snake.***
* ***Moving the Snake.***
* ***Game Over when Snake hits the boundaries.***
* ***Game Over when Snake hits it ownself***
* ***Adding the Food or Coin.***
* ***Incrementing the Length of the Snake.***
* ***Displaying the Score.***
* ***Displaying Gameover Screen***

Firstly we have created a home screen in this screen we have created a box using gotoxy ,calling writestring function and loops etc then we wrote some guidelines of the game or we can say game instructions like from which keys to move the snakes and how to exit in between game then we created difficulty level screen if using compare etc like if you press 1 hard level would be selected .

difficulty here means speed for instance hard means fast medium means normal and easy means slow .

Then we created game boundaries and a scoreboard to keep track of your progress and then finally we created a snake. the snake was initially 5 units long and then using randomize etc we created food for the snake and whenever snakes eats the food we incremented the points and snakes length and there were some problems in that i will discuss those in problem statement or code implementation section then we created a procedure named check snake to check if it touches its own body or wall and whenever it touches any of them we will simply call exit.and then scoreboard is displayed and the option to restart game is presented in front of user.

**Detailed Design and Architecture**

Assembly 8086 has been used in order to code snake game.Different components and functions were partitioned according to their necessity. Which Include:

* Box Proc: Used in the intro screen to Print a square shaped object, Which gives a more eye pleasing view to the user.
* Intro Proc: Main function for introduction screen .
* Draw Wall Proc: Used to set boundaries above which the snake can’t go. Which would result in a loss otherwise.
* DrawScoreboard Proc: Procedure used to record the points gained by the player in the game.
* ChooseSpeed Proc: Used for difficulty option. Main purpose of the function is to control the speed of the snake.
* CheckSnake Proc:Used to determine if the Snake have hit with an object such as with itself
* checkXposition: Used to check the position of the Snake.
* DrawBody: Function used to to describe the shape of the snake
* Gameover Proc: Used to instruct the compiler what to do if a snake passes the boundaries.

Similarly other functions have been used in order to describe the activity of snakes.

**Implementation ,Testing and Programing code**

I have discussed the methodology above so in this section i will discuss the main problems that we encountered during the making and implementation of this project the first main problem was if a snake is moving in upward direction and the person presses downward key or vice versa then where should snake go similarly if snake is moving leftwards and the person presses right key or vice versa then where should snake go so we made conditions by using comparisons etc to resolve this issue .another issue was how to detect if a snake touches its own body this condition took a lot of time to resolve but we eventually did it by first checking the tail of snake which was 5 units initially and then we added no of score in snake until that point and compared x and y axis and then if x and y positions were same snake dies

Now finally the last problem was to increment the length of snake the difficulty was at which position we should increment snake length like if only one unit means the tail of snake is on lets say 5th row and the whole other body was on 6th line then where should we add a unit so to resolve this issue we compared the x and y axis of tail and then added a unit to it.

so that was all from implementing and testing now the full code is below .

INCLUDE Irvine32.inc

.data

Promp11 Byte "FAST Snake Game",0

Prompt12 Byte "===============",0

xWall BYTE 52 DUP("="),0

strScore BYTE "Your score is: ",0

score BYTE 0

strTryAgain BYTE "Do you want to play again o",0

strTryAgain1 BYTE "Press 1 to Restart ",0

strTryAgain2 BYTE "Press 0 to exit",0

invalidInput BYTE "invalid input",0

gameover BYTE "game over ",0

strPoints BYTE "Score = ",0

blank BYTE " ",0

snake BYTE "o", 104 DUP("x")

xPos BYTE 45,44,43,42,41, 100 DUP(?)

yPos BYTE 15,15,15,15,15, 100 DUP(?)

xPosWall BYTE 34,34,85,85 ;position of boundaries

yPosWall BYTE 5,24,5,24

xCoinPos BYTE ?

yCoinPos BYTE ?

inputChar BYTE "+"

lastInputChar BYTE ?

strSpeed BYTE " choose Difficulty",0

strSpeed1 BYTE "1-Hard",0

strSpeed2 BYTE "2-Medium",0

strSpeed3 BYTE "3-Easy",0

speed DWORD 0

keys Byte "game guide ",0

keys1 BYTE "up = w",0

keys2 BYTE "down = s",0

keys3 BYTE "left = a",0

keys4 BYTE "right = d",0

keys5 Byte "press x to exit ",0

.code

main PROC

Call INTRO

call gamekeys

call clrscr

call ChooseSpeed ;let player to choose Speed

call clrscr

call DrawWall ;draw boundary

call DrawScoreboard ;draw scoreboard

mov esi,0

mov ecx,5

drawSnake:

call DrawPlayer ; this is snake initially 5 letters long

inc esi

loop drawSnake

call Randomize

call CreateRandomCoin

call DrawCoin ;set up finish

gameLoop:

mov dl,106 ;move cursor to coordinates

mov dh,1

call Gotoxy

; get user key input

call ReadKey

jz noKey ;jump if no key is entered

processInput:

mov bl, inputChar

mov lastInputChar, bl

mov inputChar,al ;assign variables

noKey:

cmp inputChar,"x"

je exitgame ;exit game if user input x

cmp inputChar,"w"

je checkTop

cmp inputChar,"s"

je checkBottom

cmp inputChar,"a"

je checkLeft

cmp inputChar,"d"

je checkRight

jne gameLoop ; reloop if no meaningful key was entered

; check whether can continue moving

checkBottom:

cmp lastInputChar, "w"

je dontChgDirection ;cant go down immediately after going up

mov cl, yPosWall[1]

dec cl ;one unit above the y-coordinate of the lower bound

cmp yPos[0],cl

jl moveDown

je died ;die if crash into the wall

checkLeft:

cmp lastInputChar, "+" ;check whether its the start of the game

je dontGoLeft

cmp lastInputChar, "d"

je dontChgDirection

mov cl, xPosWall[0]

inc cl

cmp xPos[0],cl

jg moveLeft

je died ; check for left

checkRight:

cmp lastInputChar, "a"

je dontChgDirection

mov cl, xPosWall[2]

dec cl

cmp xPos[0],cl

jl moveRight

je died ; check for right

checkTop:

cmp lastInputChar, "s"

je dontChgDirection

mov cl, yPosWall[0]

inc cl

cmp yPos,cl

jg moveUp

je died ; check for up

moveUp:

mov eax, speed ;slow down the moving

add eax, speed

call delay

mov esi, 0 ;index 0(snake head)

call UpdatePlayer

mov ah, yPos[esi]

mov al, xPos[esi] ;al ah stores the pos of the snake's next unit

dec yPos[esi] ;move the head up

call DrawPlayer

call DrawBody

call CheckSnake

moveDown: ;move down

mov eax, speed

add eax, speed

call delay

mov esi, 0

call UpdatePlayer

mov ah, yPos[esi]

mov al, xPos[esi]

inc yPos[esi]

call DrawPlayer

call DrawBody

call CheckSnake

moveLeft: ;move left

mov eax, speed

call delay

mov esi, 0

call UpdatePlayer

mov ah, yPos[esi]

mov al, xPos[esi]

dec xPos[esi]

call DrawPlayer

call DrawBody

call CheckSnake

moveRight: ;move right

mov eax, speed

call delay

mov esi, 0

call UpdatePlayer

mov ah, yPos[esi]

mov al, xPos[esi]

inc xPos[esi]

call DrawPlayer

call DrawBody

call CheckSnake

; getting points

checkcoin::

mov esi,0

mov bl,xPos[0]

cmp bl,xCoinPos

jne gameloop ;reloop if snake is not intersecting with coin

mov bl,yPos[0]

cmp bl,yCoinPos

jne gameloop ;reloop if snake is not intersecting with coin

call EatingCoin ;call to update score, append snake and generate new coin

jmp gameLoop ;reiterate the gameloop

dontChgDirection: ;dont allow user to change direction

mov inputChar, bl ;set current inputChar as previous

jmp noKey ;jump back to continue moving the same direction

dontGoLeft: ;forbids the snake to go left at the begining of the game

mov inputChar, "+" ;set current inputChar as "+"

jmp gameLoop ;restart the game loop

died::

call gameover1

playagn::

call StartofGame ;reinitialise everything

exitgame::

call clrscr

exit

INVOKE ExitProcess,0

main ENDP

Box PROC

mov ecx,80

mov dl,10 ; dl = x-axis column

; dh = y-axis row

mov dh,4 ;row

LOOPS:

call gotoxy

mov al,61 ;Ascii

call writechar

mov eax , 3

call delay

inc dl

LOOP LOOPS

mov ecx,20

dec dl

inc dh

LOOPSS:

call gotoxy

mov al,124

call writechar

mov eax , 3

call delay

inc dh

LOOP LOOPSS

mov ecx,80

LOOPPSS:

call gotoxy

mov al,61

call writechar

mov eax , 3

call delay

dec dl

LOOP LOOPPSS

mov ecx,20

inc dl

dec dh

LOOPPSSS:

call gotoxy

mov al,124

call writechar

mov eax , 3

call delay

dec dh

LOOP LOOPPSSS

ret

Box endp

Intro Proc

Call Clrscr

Call Box

mov eax,10

call SetTextColor

mov dl,40

mov dh,13

call gotoxy

mov edx,offset promp11

call writestring

mov eax,14

call SetTextColor

mov dl,40

mov dh,14

call gotoxy

mov edx,offset prompt12

call writestring

mov eax,15

call SetTextColor

mov ecx,13

CR:

call crlf

LOOP CR

call waitmsg ;to print the message and pause the program until enter is pressed

ret

Intro Endp

DrawWall PROC ;procedure to draw wall

mov dl,xPosWall[0]

mov dh,yPosWall[0]

call Gotoxy

mov edx,OFFSET xWall

call WriteString ;draw upper wall

mov dl,xPosWall[1]

mov dh,yPosWall[1]

call Gotoxy

mov edx,OFFSET xWall

call WriteString ;draw lower wall

mov dl, xPosWall[2]

mov dh, yPosWall[2]

mov eax,"|"

inc yPosWall[3]

L11:

call Gotoxy

call WriteChar

inc dh

cmp dh, yPosWall[3] ;draw right wall

jl L11

mov dl, xPosWall[0]

mov dh, yPosWall[0]

mov eax,"|"

L12:

call Gotoxy

call WriteChar

inc dh

cmp dh, yPosWall[3] ;draw left wall

jl L12

ret

DrawWall ENDP

DrawScoreboard PROC ;procedure to draw scoreboard

mov dl,2 ;column

mov dh,1 ;row

call Gotoxy

mov edx,OFFSET strScore ;print string that indicates score

call WriteString

mov eax,"0"

call WriteChar ;scoreboard starts with 0

ret

DrawScoreboard ENDP

gamekeys proc

mov eax,15

call SetTextColor

mov dl,40

mov dh,15

call Gotoxy

mov edx,OFFSET keys ; prompt to enter integers (1,2,3)

call WriteString

mov eax,14

call SetTextColor

mov dl,40

mov dh,16

call Gotoxy

mov edx,OFFSET keys1 ; prompt to enter integers (1,2,3)

call WriteString

mov eax,14

call SetTextColor

mov dl,40

mov dh,17

call Gotoxy

mov edx,OFFSET keys2 ; prompt to enter integers (1,2,3)

call WriteString

mov eax,14

call SetTextColor

mov dl,40

mov dh,18

call Gotoxy

mov edx,OFFSET keys3 ; prompt to enter integers (1,2,3)

call WriteString

mov eax,14

call SetTextColor

mov dl,40

mov dh,19

call Gotoxy

mov edx,OFFSET keys4 ; prompt to enter integers (1,2,3)

call WriteString

mov eax,15

call SetTextColor

mov dl,40

mov dh,20

call Gotoxy

mov edx,OFFSET keys5 ; prompt to enter integers (1,2,3)

call WriteString

mov ecx,13

R:

call crlf

LOOP R

call waitmsg

ret

gamekeys ENDP

ChooseSpeed PROC ;procedure for player to choose speed

mov edx,0

mov dl,45

mov dh,11

call Gotoxy

mov edx,OFFSET strSpeed ; prompt to enter integers (1,2,3)

call WriteString

mov dl,45

mov dh,13

call Gotoxy

mov edx,OFFSET strSpeed1 ; prompt to enter integers (1,2,3)

call WriteString

mov dl,45

mov dh,15

call Gotoxy

mov edx,OFFSET strSpeed2 ; prompt to enter integers (1,2,3)

call WriteString

mov dl,45

mov dh,17

call Gotoxy

mov edx,OFFSET strSpeed3 ; prompt to enter integers (1,2,3)

call WriteString

call crlf

mov esi, 40 ; milisecond difference per speed level

mov eax,0

call readInt

cmp ax,1 ;input validation

jl invalidspeed

cmp ax, 3

jg invalidspeed

mul esi

mov speed, eax ;assign speed variable in mililiseconds

ret

invalidspeed: ;jump here if user entered an invalid number

mov dl,105

mov dh,1

call Gotoxy

mov edx, OFFSET invalidInput ;print error message

call WriteString

mov ax, 1500

call delay

mov dl,105

mov dh,1

call Gotoxy

mov edx, OFFSET blank ;erase error message after 1.5 secs of delay

call writeString

call ChooseSpeed ;call procedure for user to choose again

ret

ChooseSpeed ENDP

DrawPlayer PROC ; draw player at (xPos,yPos)

mov dl,xPos[esi]

mov dh,yPos[esi]

call Gotoxy

mov dl, al ;temporarily save al in dl

mov al, snake[esi]

call WriteChar

mov al, dl

ret

DrawPlayer ENDP

UpdatePlayer PROC ; erase player at (xPos,yPos)

mov dl, xPos[esi]

mov dh,yPos[esi]

call Gotoxy

mov dl, al ;temporarily save al in dl

mov al, " "

call WriteChar

mov al, dl

ret

UpdatePlayer ENDP

DrawCoin PROC ;procedure to draw coin

mov eax,yellow (yellow \* 16)

call SetTextColor ;set color to yellow for coin

mov dl,xCoinPos

mov dh,yCoinPos

call Gotoxy

mov al,"X"

call WriteChar

mov eax,white (black \* 16) ;reset color to black and white

call SetTextColor

ret

DrawCoin ENDP

CreateRandomCoin PROC ;procedure to create a random coin

mov eax,49

call RandomRange ;0-49

add eax, 35 ;35-84

mov xCoinPos,al

mov eax,17

call RandomRange ;0-17

add eax, 6 ;6-23

mov yCoinPos,al

mov ecx, 5

add cl, score ;loop number of snake unit

mov esi, 0

checkCoinXPos:

movzx eax, xCoinPos

cmp al, xPos[esi]

je checkCoinYPos ;jump if xPos of snake at esi = xPos of coin

continueloop:

inc esi

loop checkCoinXPos

ret ; return when coin is not on snake

checkCoinYPos:

movzx eax, yCoinPos

cmp al, yPos[esi]

jne continueloop ; jump back to continue loop if yPos of snake at esi != yPos of coin

call CreateRandomCoin ; coin generated on snake, calling function again to create another set of coordinates

CreateRandomCoin ENDP

CheckSnake PROC ;for checking if it touches his own body

mov al, xPos[0]

mov ah, yPos[0]

mov esi,4 ;start checking from index 4(5th unit)

mov ecx,1

add cl,score

checkXposition:

cmp xPos[esi], al ;check if xpos same or not

je XposSame

contloop:

inc esi

loop checkXposition

jmp checkcoin

XposSame: ; if xpos same, check for y pos

cmp yPos[esi], ah

je died ;if collides, snake dies

jmp contloop

CheckSnake ENDP

DrawBody PROC ;procedure to print body of the snake

mov ecx, 4

add cl, score ;number of iterations to print the snake body n tail

printbodyloop:

inc esi ;loop to print remaining part of snake

call UpdatePlayer

mov dl, xPos[esi]

mov dh, yPos[esi] ;dl dh temporarily stores the current pos

mov yPos[esi], ah

mov xPos[esi], al ;assign new position

mov al, dl

mov ah,dh ;move the current position back into al ah

call DrawPlayer

cmp esi, ecx

jl printbodyloop

ret

DrawBody ENDP

EatingCoin PROC

inc score

mov ebx,4

add bl, score

mov esi, ebx

mov ah, yPos[esi-1]

mov al, xPos[esi-1]

mov xPos[esi], al ;add one letter x to the snake

mov yPos[esi], ah ;pos of new tail = pos of old tail

cmp xPos[esi-2], al ;check if the old tail and the unit before is on the yAxis

jne checky ;jump if not on the yAxis

cmp yPos[esi-2], ah ;check if the new tail should be above or below of the old tail

jl incy

jg decy

incy: ;inc if below

inc yPos[esi]

jmp continue

decy: ;dec if above

dec yPos[esi]

jmp continue

checky: ;old tail and the unit before is on the xAxis

cmp yPos[esi-2], ah ;check if the new tail should be right or left of the old tail

jl incx

jg decx

incx: ;inc if right

inc xPos[esi]

jmp continue

decx: ;dec if left

dec xPos[esi]

continue: ;add snake tail and update new coin

call DrawPlayer

call CreateRandomCoin

call DrawCoin

mov dl,17 ; write updated score

mov dh,1

call Gotoxy

mov al,score

call WriteInt

ret

EatingCoin ENDP

gameover1 PROC

mov eax, 1000

call delay

Call ClrScr

call Box

mov eax,11

call SetTextColor

mov dl, 40

mov dh, 13

call Gotoxy

mov edx, OFFSET gameover ;"game over"

call WriteString

mov dl, 40

mov dh, 14

call Gotoxy

mov edx, OFFSET strPoints ;display score

call WriteString

movzx eax, score

call WriteDec

mov dl, 40

mov dh, 15

call Gotoxy

mov edx, OFFSET strTryAgain

call WriteString ;"try again?"

mov dl, 40

mov dh, 16

call Gotoxy

mov edx, OFFSET strTryAgain1

call WriteString

mov dl, 40

mov dh, 17

call Gotoxy

mov edx, OFFSET strTryAgain2

call WriteString

retry:

mov dh, 19

mov dl, 56

call Gotoxy

call ReadInt ;get user input

cmp al, 1

je playagn ;playagn

cmp al, 0

je exitgame ;exitgame

mov dh, 17

call Gotoxy

mov edx, OFFSET invalidInput ;"Invalid input"

call WriteString

mov dl, 56

mov dh, 19

call Gotoxy

mov edx, OFFSET blank ;erase previous input

call WriteString

jmp retry ;let user input again

gameover1 ENDP

StartofGame PROC ;procedure to reinitialize everything

mov xPos[0], 45

mov xPos[1], 44

mov xPos[2], 43

mov xPos[3], 42

mov xPos[4], 41

mov yPos[0], 15

mov yPos[1], 15

mov yPos[2], 15

mov yPos[3], 15

mov yPos[4], 15 ;reinitialize snake position

mov score,0 ;reinitialize score

mov lastInputChar, 0

mov inputChar, "+" ;reinitialize inputChar and lastInputChar

dec yPosWall[3] ;reset wall position

Call ClrScr

jmp main ;start over the game

StartofGame ENDP

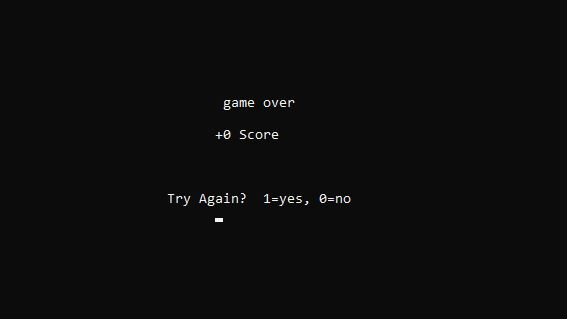
END main

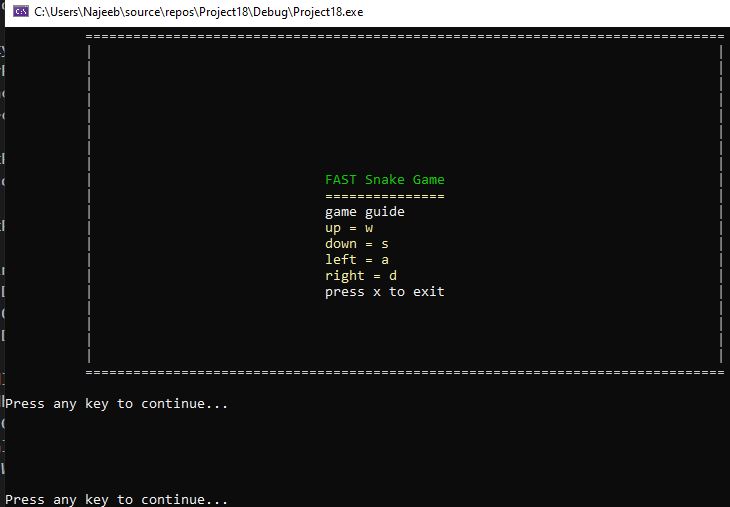
**Results:**

1. At the start of the program the player is asked to press any key to start the program.

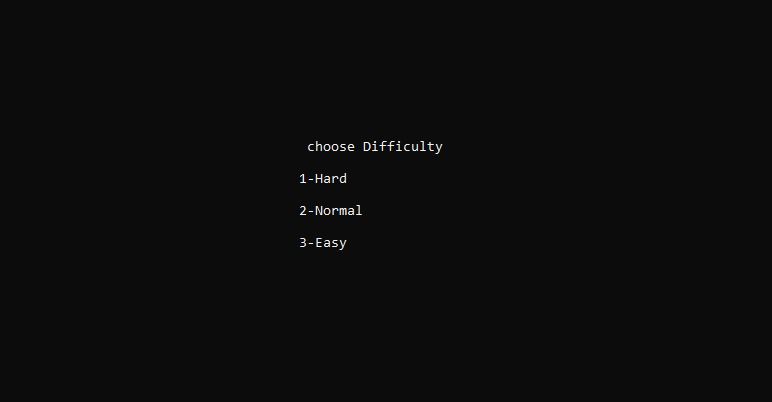


2) Game guide is to show the user how the program works. Press any key to continue

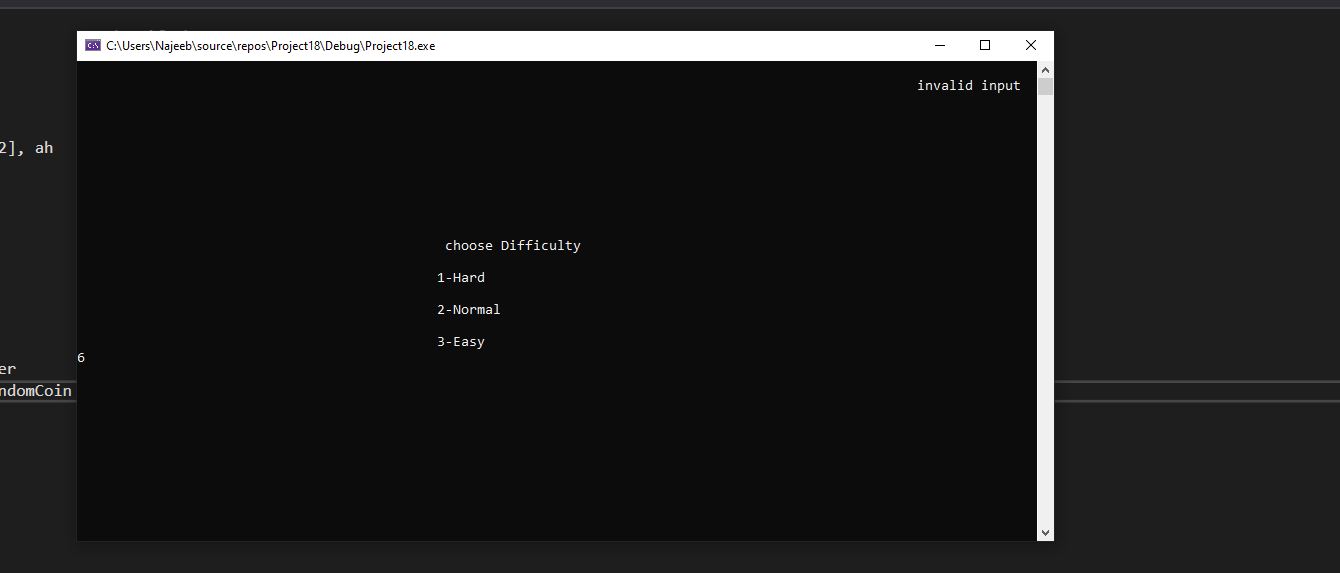




3) User is asked to choose between 3 difficulties



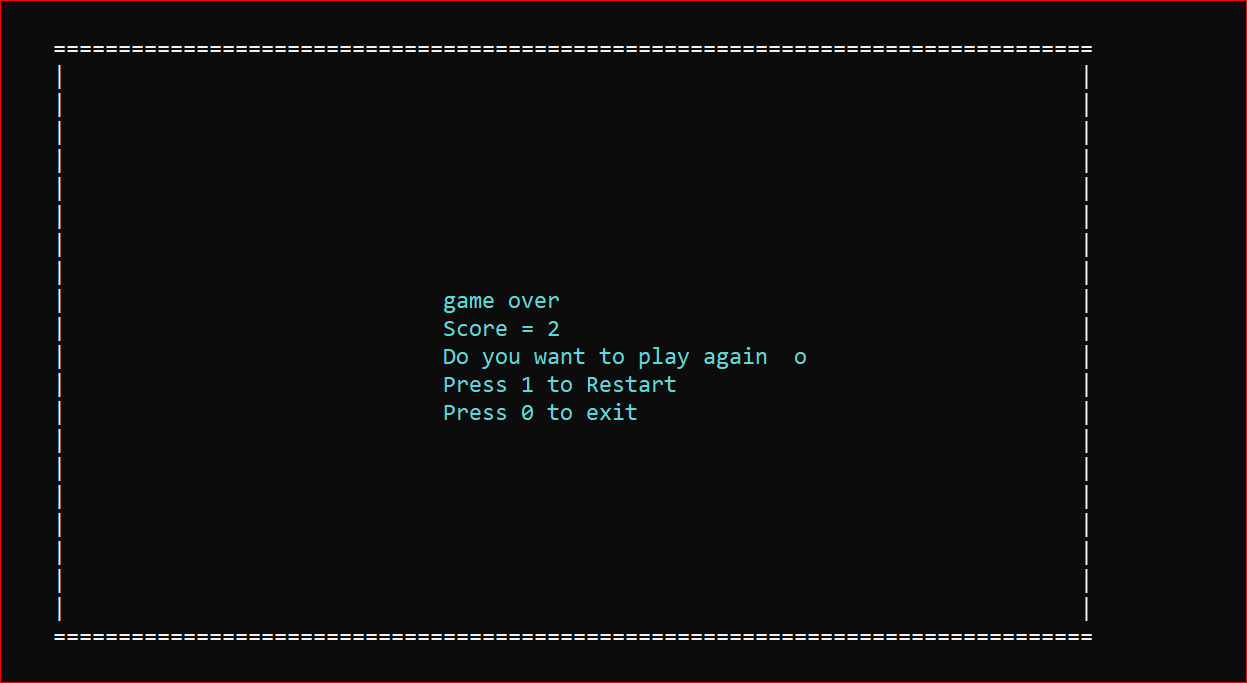
3)(b) If a invalid input is given such as 6 as shown then invalid input is displayed at top right of the screen.



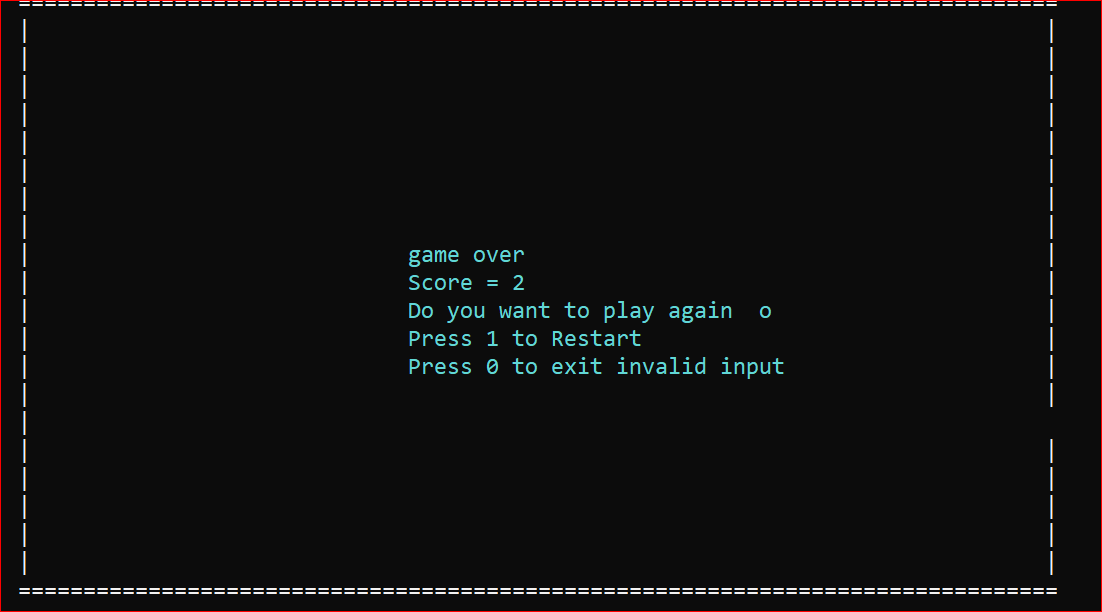
4) User plays the game. Any input other than the keys shown on the game guide are ignored.



5) User is asked if he wants to continue the game



5)(b) If an invalid key is entered then invalid input is shown.

****

**Conclusion:**

Snake game is a computer action game, whose goal is to control a snake to move and collect food in a map. In this project we have developed a controller based on movement rating functions considering smoothness, space, and food.Snake is a classic game that has been played for decades. It can also be seen as a good practice program for someone who wants to join video gaming industry,

We have used most of the concepts taught throughout the semester in this project. Which have helped us tremendously to improve our assembly programming skills.