Lab: 13

Fahid Imran

Roll No: 23i-0061

COAL

Instructor

Mr. Sulaman Saboor

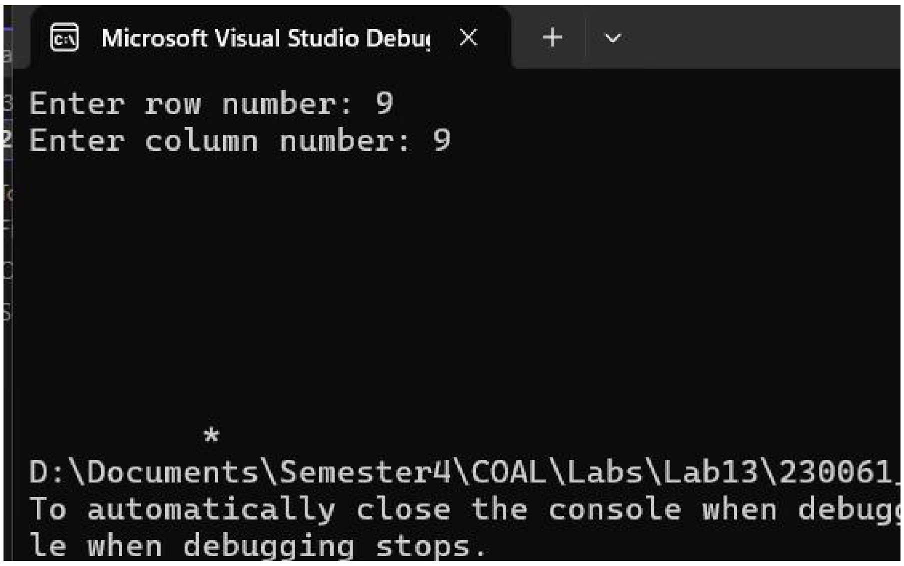
Fast NUCES Islamabad Campus

Task1:

Code:

```
include irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte '*', 0
msg1 byte 'Enter row number: ', 0
msg2 byte 'Enter column number: ', 0
x_cord byte ?
y_cord byte ?
.code
main PROC
  mov edx, offset msg1
  call writestring
  call readint
  mov x_cord, al

  mov edx, offset msg2
  call writestring
  call readint
  mov y_cord, al

  mov dl, x_cord
  mov dh, y_cord
  call gotoxy
  mov edx, offset msg
  call writestring

exit
main endp
end main
```

Output:



```
Enter row number: 9
Enter column number: 9





        *
D:\Documents\Semester4\COAL\Labs\Lab13\230061
To automatically close the console when debugg
le when debugging stops.
```

Task2:

Code:

```
include irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte '*', 0
msg1 byte 'Enter row number: ', 0
msg2 byte 'Enter column number: ', 0
msg3 byte 'Enter text color: ', 0
msg4 byte 'Enter back color: ', 0
color0 byte 'Enter 0 for BLACK', 0
color1 byte 'Enter 1 for BLUE', 0
color2 byte 'Enter 2 for GREEN', 0
color3 byte 'Enter 3 for CYAN', 0
color4 byte 'Enter 4 for RED', 0
color5 byte 'Enter 5 for MAGENTA', 0
color6 byte 'Enter 6 for BROWN', 0
color7 byte 'Enter 7 for LIGHT GRAY', 0
color8 byte 'Enter 8 for DARK GRAY', 0
color9 byte 'Enter 9 for LIGHT BLUE', 0
color10 byte 'Enter 10 for LIGHT GREEN', 0
color11 byte 'Enter 11 for LIGHT CYAN', 0
```

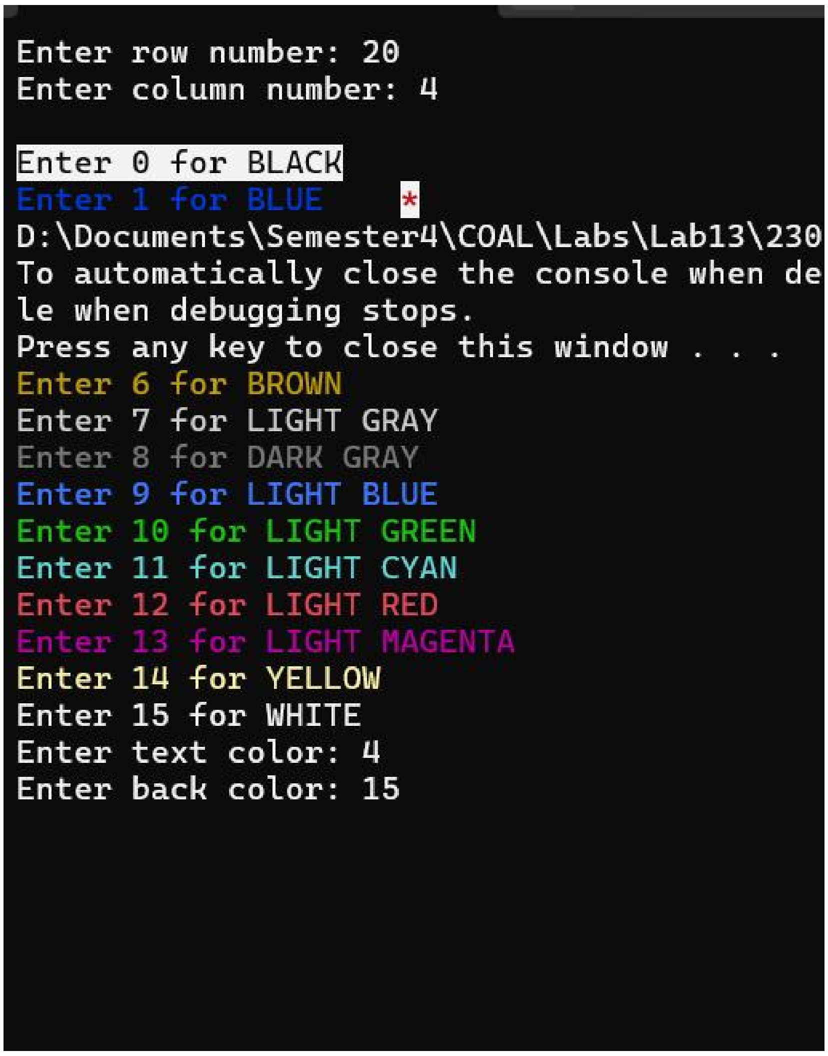
```
color12 byte 'Enter 12 for LIGHT RED', 0
color13 byte 'Enter 13 for LIGHT MAGENTA', 0
color14 byte 'Enter 14 for YELLOW', 0
color15 byte 'Enter 15 for WHITE', 0

x_cord byte ?
y_cord byte ?
text_color dword ?
back_color dword ?
.code
main PROC


  mov edx, offset msg1
  call writestring
  call readint
  mov x_cord, al

  mov edx, offset msg2
  call writestring
  call readint
  mov y_cord, al

  call crlf

  mov eax, 0 + (15*16)
  call SetTextColor
  mov edx, offset color0
  call writestring

  call crlf

  mov eax, 1
  call SetTextColor
  mov edx, offset color1
  call writestring

  call crlf

  mov eax, 2
  call SetTextColor
  mov edx, offset color2
```

```
call writestring

call crlf

mov eax, 3
call SetTextColor
mov edx, offset color3
call writestring
call crlf

mov eax, 4
call SetTextColor
mov edx, offset color4
call writestring
call crlf

mov eax, 5
call SetTextColor
mov edx, offset color5
call writestring
call crlf

mov eax, 6
call SetTextColor
mov edx, offset color6
call writestring
call crlf

mov eax, 7
call SetTextColor
mov edx, offset color7
call writestring
call crlf

mov eax, 8
call SetTextColor
mov edx, offset color8
call writestring
call crlf

mov eax, 9
call SetTextColor
```

```
mov edx, offset color9
call writestring
call crlf

mov eax, 10
call SetTextColor
mov edx, offset color10
call writestring
call crlf

mov eax, 11
call SetTextColor
mov edx, offset color11
call writestring
call crlf

mov eax, 12
call SetTextColor
mov edx, offset color12
call writestring
call crlf

mov eax, 13
call SetTextColor
mov edx, offset color13
call writestring
call crlf

mov eax, 14
call SetTextColor
mov edx, offset color14
call writestring
call crlf

mov eax, 15
call SetTextColor
mov edx, offset color15
call writestring
call crlf

mov edx, offset msg3
call writestring
```

```
        call readint
        mov text_color, eax

        mov edx, offset msg4
        call writestring
        call readint
        mov back_color, eax
        call crlf


        mov eax,back_color
        mov ebx, 16
        mul ebx
        add eax, text_color
        call SetTextColor







        mov dl, x_cord
        mov dh, y_cord
        call gotoxy
        mov edx, offset msg
        call writestring

        mov eax,15 +(0 * 16 )
        call SetTextColor

      exit
      main endp
  end main
```

Output:



```
Enter row number: 20
Enter column number: 4

Enter 0 for BLACK
Enter 1 for BLUE      *
D:\Documents\Semester4\COAL\Labs\Lab13\230
To automatically close the console when de
le when debugging stops.
Press any key to close this window . . .
Enter 6 for BROWN
Enter 7 for LIGHT GRAY
Enter 8 for DARK GRAY
Enter 9 for LIGHT BLUE
Enter 10 for LIGHT GREEN
Enter 11 for LIGHT CYAN
Enter 12 for LIGHT RED
Enter 13 for LIGHT MAGENTA
Enter 14 for YELLOW
Enter 15 for WHITE
Enter text color: 4
Enter back color: 15
```

Task3:

Code:

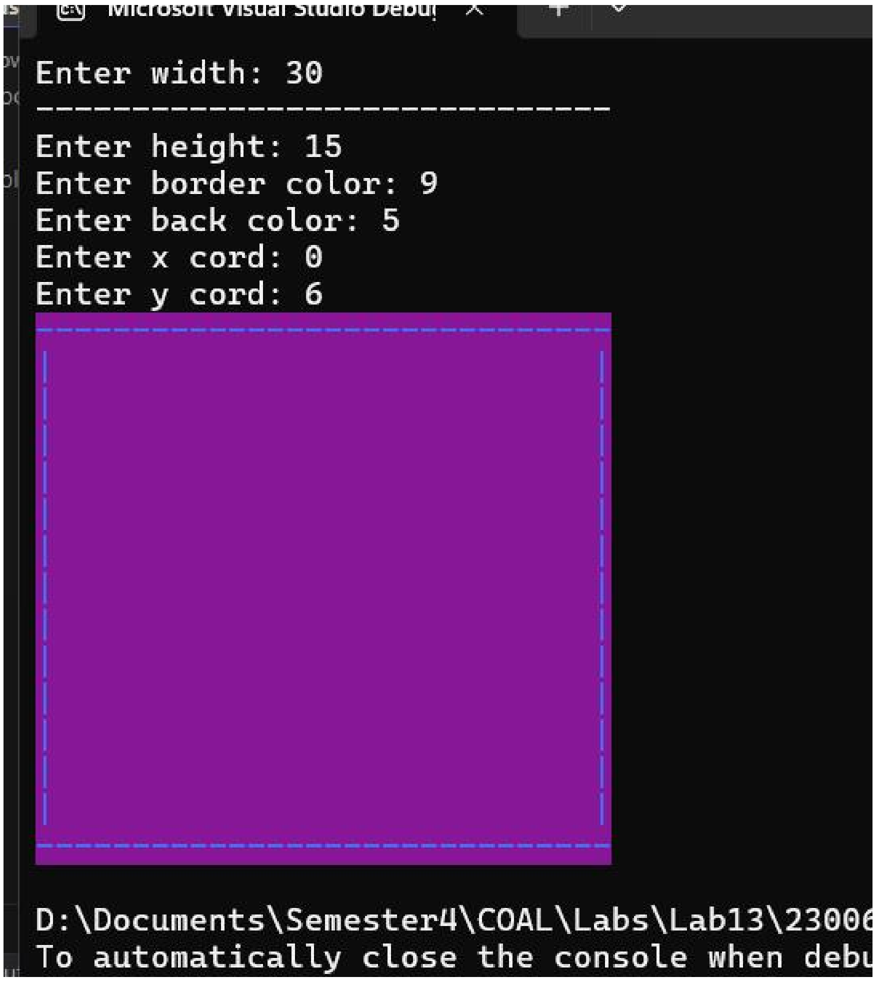
```
include irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte '*', 0
msg1 byte 'Enter width: ', 0
msg2 byte 'Enter height: ', 0
msg3 byte 'Enter border color: ', 0
msg4 byte 'Enter back color: ', 0
msg5 byte 'Enter x cord: ', 0
```

```
msg6 byte 'Enter y cord: ', 0

color0 byte 'Enter 0 for BLACK', 0
color1 byte 'Enter 1 for BLUE', 0
color2 byte 'Enter 2 for GREEN', 0
color3 byte 'Enter 3 for CYAN', 0
color4 byte 'Enter 4 for RED', 0
color5 byte 'Enter 5 for MAGENTA', 0
color6 byte 'Enter 6 for BROWN', 0
color7 byte 'Enter 7 for LIGHT GRAY', 0
color8 byte 'Enter 8 for DARK GRAY', 0
color9 byte 'Enter 9 for LIGHT BLUE', 0
color10 byte 'Enter 10 for LIGHT GREEN', 0
color11 byte 'Enter 11 for LIGHT CYAN', 0
color12 byte 'Enter 12 for LIGHT RED', 0
color13 byte 'Enter 13 for LIGHT MAGENTA', 0
color14 byte 'Enter 14 for YELLOW', 0
color15 byte 'Enter 15 for WHITE', 0

wall byte '|', 0
arr byte 100 dup (0)
arr2 byte 100 dup (0)

box_width byte ?
box_height byte ?
border_color dword ?
back_color dword ?
x_cord byte ?
y_cord byte ?


.code
main PROC


  mov edx, offset msg1
  call writestring
  call readint
  mov box_width, al

  mov ecx, eax
  mov esi, 0
```

```
label1:

      mov byte ptr [offset arr + esi], '-'
      mov byte ptr [offset arr2 + esi], ' '

      inc esi

loop label1

mov byte ptr [offset arr2 + 0], '|'
dec eax
mov byte ptr [offset arr2 + eax], '|'

mov edx, offset arr
call writestring
call crlf

mov edx, offset msg2
call writestring
call readint
mov box_height, al

mov edx, offset msg3
call writestring
call readint
mov border_color, eax

mov edx, offset msg4
call writestring
call readint
mov back_color, eax

mov edx, offset msg5
call writestring
call readint
mov x_cord, al

mov edx, offset msg6
call writestring
call readint
mov y_cord, al
```

```
        mov dl , x_cord
        mov dh, y_cord
        mov eax, back_color
        mov ebx, 16
        mul ebx
        add eax, border_color

        call setTextColor

        mov edx, offset arr
        call writestring

        mov ecx, 0
        mov cl, box_height
        dec ecx
        dec ecx
        call crlf
        label2:
              mov edx, offset arr2
              call writestring
        call crlf

        loop label2

        mov edx, offset arr
        call writestring
        call crlf

        mov eax , 15 + (0*16)
        call setTextColor




      exit
      main endp
end main
```

Output:



```
Enter width: 30
------------------------------
Enter height: 15
Enter border color: 9
Enter back color: 5
Enter x cord: 0
Enter y cord: 6
```

```
D:\Documents\Semester4\COAL\Labs\Lab13\23006
To automatically close the console when debu
```

Task4:

Code:

```
INCLUDE Irvine32.inc
.386
.model flat, stdcall
.stack 4096
.data
ground BYTE "-------------------------------------------------------------------------------
-------------------------------------",0
strScore BYTE "Your score is: ",0
score BYTE 0
xPos BYTE 20
yPos BYTE 20
```

```
xCoinPos BYTE ?
yCoinPos BYTE ?
inputChar BYTE ?
direction BYTE 0      ; 0=right, 1=down, 2=left, 3=up
snakeBody BYTE 100 DUP(0,0)    ; x,y pairs for snake segments
snakeLength BYTE 3             ; starting snake length
.code
main PROC
   call Clrscr
   mov dl,0
   mov dh,29
   call Gotoxy
   mov edx,OFFSET ground
   call WriteString

   call InitSnake
   call CreateRandomCoin
   call DrawCoin
   call Randomize

   gameLoop:
      mov eax,white + (black * 16)
      call SetTextColor

      mov dl,0
      mov dh,0
      call Gotoxy
      mov edx,OFFSET strScore
      call WriteString
      movzx eax,score
      call WriteInt

      call CheckCoinCollision
      call ProcessInput
```

```
        call MoveSnake
        call DrawSnake

        mov eax,100       ; Control game speed
        call Delay
    jmp gameLoop

    exit
main ENDP

InitSnake PROC
    mov ecx,0
    movzx ecx,snakeLength
    mov esi,0

    initLoop:
        mov al,xPos
        sub al,cl       ; Place segments to the left of head
        mov snakeBody[esi],al
        inc esi
        mov al,yPos
        mov snakeBody[esi],al
        inc esi
        loop initLoop
    ret
InitSnake ENDP

DrawSnake PROC
    mov eax,green + (black * 16)
    call SetTextColor

    movzx ecx,snakeLength
    mov esi,0
    drawLoop:
        mov dl,snakeBody[esi]    ; x position
        mov dh,snakeBody[esi+1]  ; y position
        call Gotoxy

        cmp esi,0
        jne bodySegment
        mov al,"O"              ; Draw head as O
        jmp drawChar
    bodySegment:
        mov al,"o"              ; Draw body as o
    drawChar:
        call WriteChar
        add esi,2
        loop drawLoop
    ret
DrawSnake ENDP

UpdateSnake PROC
    movzx ecx,snakeLength
    mov esi,0

    updateLoop:
        mov dl,snakeBody[esi]
        mov dh,snakeBody[esi+1]
        call Gotoxy
        mov al," "
        call WriteChar
        add esi,2
        loop updateLoop
    ret
UpdateSnake ENDP

MoveSnake PROC
```

```
    call UpdateSnake

    mov al,snakeLength
    dec al
    movzx ecx,al          ; Move all segments except head

    moveLoop:
        mov esi,ecx
        shl esi,1         ; Convert to byte offset

        mov al,snakeBody[esi-2]  ; Copy position from segment ahead
        mov snakeBody[esi],al
        mov al,snakeBody[esi-1]
        mov snakeBody[esi+1],al
        loop moveLoop

    cmp direction,0       ; Right
    jne checkDown
    inc snakeBody[0]
    jmp boundaryCheck

checkDown:
    cmp direction,1       ; Down
    jne checkLeft
    inc snakeBody[1]
    jmp boundaryCheck

checkLeft:
    cmp direction,2       ; Left
    jne checkUp
    dec snakeBody[0]
    jmp boundaryCheck

checkUp:
    cmp direction,3        ; Up
    jne boundaryCheck
    dec snakeBody[1]

boundaryCheck:
    cmp snakeBody[0],0     ; Left boundary
    jl wrapRight
    cmp snakeBody[0],79    ; Right boundary
    jg wrapLeft
    cmp snakeBody[1],0     ; Top boundary
    jl wrapBottom
    cmp snakeBody[1],28    ; Bottom boundary (above ground)
    jl doneMove

    mov snakeBody[1],27    ; Keep on ground
    jmp doneMove

wrapRight:
    mov snakeBody[0],79
    jmp doneMove
wrapLeft:
    mov snakeBody[0],0
    jmp doneMove
wrapBottom:
    mov snakeBody[1],27

doneMove:
    mov al,snakeBody[0]    ; Update xPos and yPos to match head
    mov xPos,al
    mov al,snakeBody[1]
    mov yPos,al
    ret
MoveSnake ENDP
```

```
CheckCoinCollision PROC
    mov bl,snakeBody[0]    ; Head x position
    cmp bl,xCoinPos
    jne notCollecting
    mov bl,snakeBody[1]    ; Head y position
    cmp bl,yCoinPos
    jne notCollecting

    inc score
    inc snakeLength        ; Grow snake

    call CreateRandomCoin
    call DrawCoin

notCollecting:
    ret
CheckCoinCollision ENDP

ProcessInput PROC
    mov eax,50             ; Check for key press with short delay
    call Delay

    call ReadKey           ; AL = ascii code, Non-blocking
    jz noKey               ; ZF=1 means no key pressed

    mov inputChar,al

    cmp inputChar,"x"      ; Exit game
    je exitGame
    cmp inputChar,"w"      ; Up
    jne checkDown
    mov direction,3
    jmp inputDone
```

```
checkDown:
    cmp inputChar,"s"      ; Down
    jne checkLeft
    mov direction,1
    jmp inputDone

checkLeft:
    cmp inputChar,"a"      ; Left
    jne checkRight
    mov direction,2
    jmp inputDone

checkRight:
    cmp inputChar,"d"      ; Right
    jne inputDone
    mov direction,0

inputDone:
    ret

noKey:
    ret

exitGame:
    exit                   ; Exit program if x is pressed
ProcessInput ENDP

DrawCoin PROC
    mov eax,yellow + (black * 16)
    call SetTextColor
    mov dl,xCoinPos
    mov dh,yCoinPos
    call Gotoxy
    mov al,"$"             ; Changed coin symbol to $
    call WriteChar
    ret
DrawCoin ENDP

CreateRandomCoin PROC
    mov eax,75              ; Random x between 0-75
    call RandomRange
    mov xCoinPos,al

    mov eax,26              ; Random y between 0-26
    call RandomRange
    mov yCoinPos,al
    ret
CreateRandomCoin ENDP

END main
```

Output: