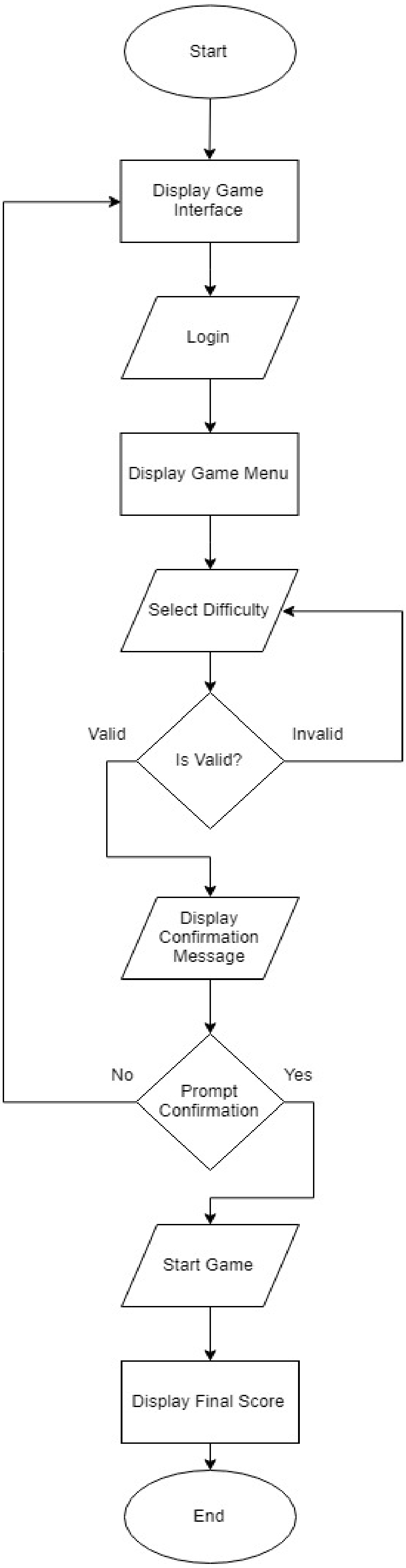
System Title: FireFly Snake Game Vers.1

FlowChart:



User Guides:

Step 1: Execute the project.

Step 2: Enter your name once you jump to login page.

Step 3: Select your difficulty that provided in the game menu.

Step 4: Confirm your selection.

Step 5: Enjoy the game.

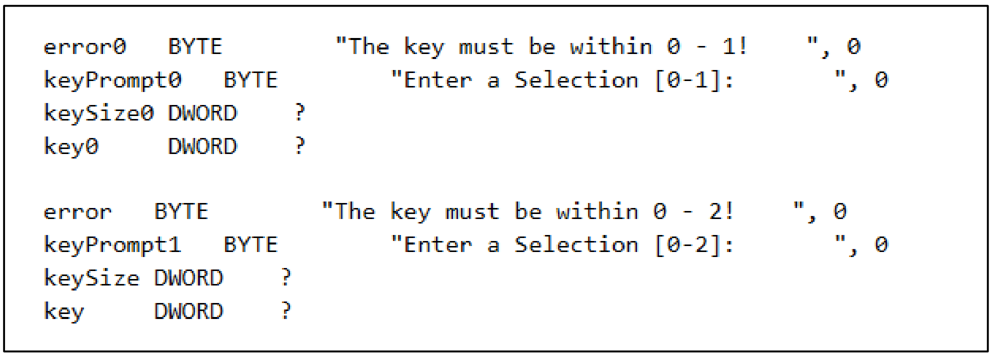
Step 6: View your gameplay final score.

Description of Each Module:

Pua Kok Bin

Module D:

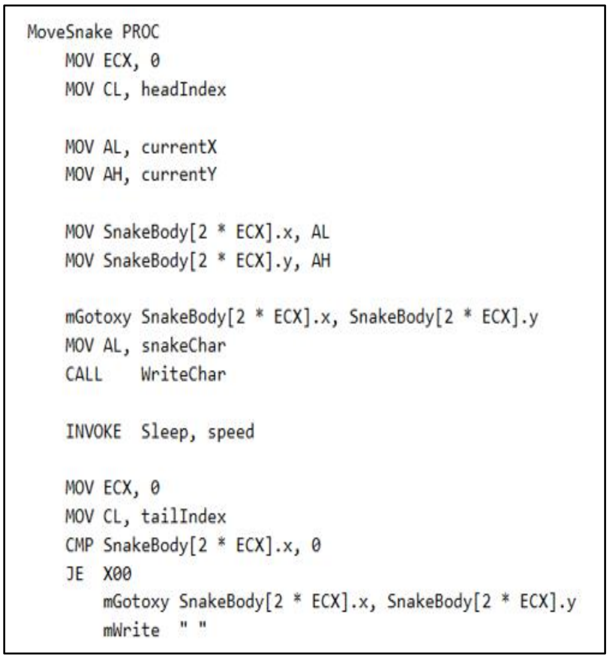
Description: To validate user input, wrong input with re-prompt the message.



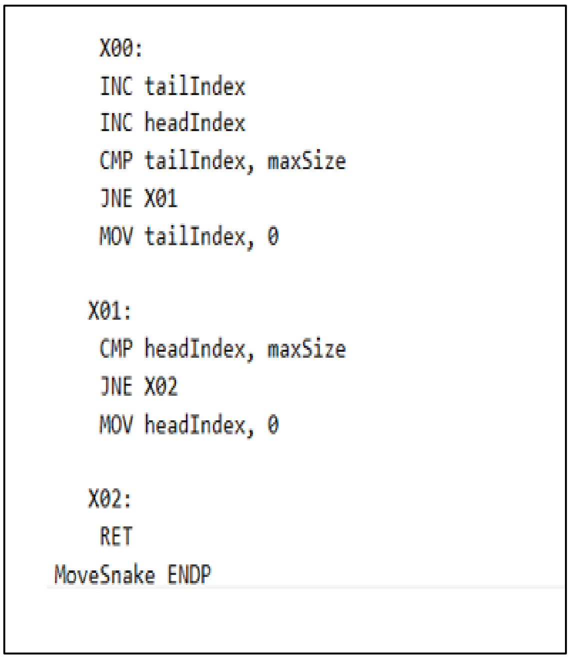
```
error0    BYTE        "The key must be within 0 - 1!    ", 0
keyPrompt0    BYTE        "Enter a Selection [0-1]:       ", 0
keySize0 DWORD    ?
key0     DWORD    ?

error    BYTE        "The key must be within 0 - 2!    ", 0
keyPrompt1    BYTE        "Enter a Selection [0-2]:       ", 0
keySize DWORD    ?
key     DWORD    ?
```

Module Move Snake:

Description: To move snake within the wall based on x,y coordinate given.



```
MoveSnake PROC
    MOV ECX, 0
    MOV CL, headIndex

    MOV AL, currentX
    MOV AH, currentY

    MOV SnakeBody[2 * ECX].x, AL
    MOV SnakeBody[2 * ECX].y, AH

    mGotoxy SnakeBody[2 * ECX].x, SnakeBody[2 * ECX].y
    MOV AL, snakeChar
    CALL    WriteChar

    INVOKE  Sleep, speed

    MOV ECX, 0
    MOV CL, tailIndex
    CMP SnakeBody[2 * ECX].x, 0
    JE  X00
        mGotoxy SnakeBody[2 * ECX].x, SnakeBody[2 * ECX].y
        mWrite  " "
```

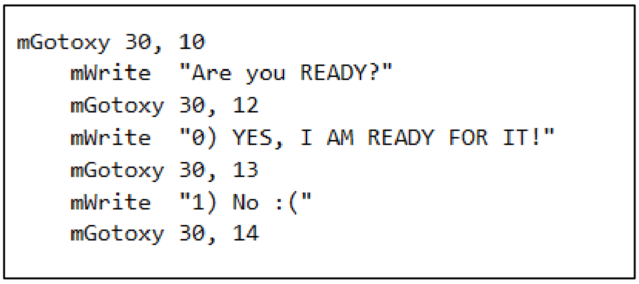


```
    X00:
    INC tailIndex
    INC headIndex
    CMP tailIndex, maxSize
    JNE X01
    MOV tailIndex, 0

   X01:
    CMP headIndex, maxSize
    JNE X02
    MOV headIndex, 0

   X02:
    RET
MoveSnake ENDP
```

Leong Zheng Jack

Module C:

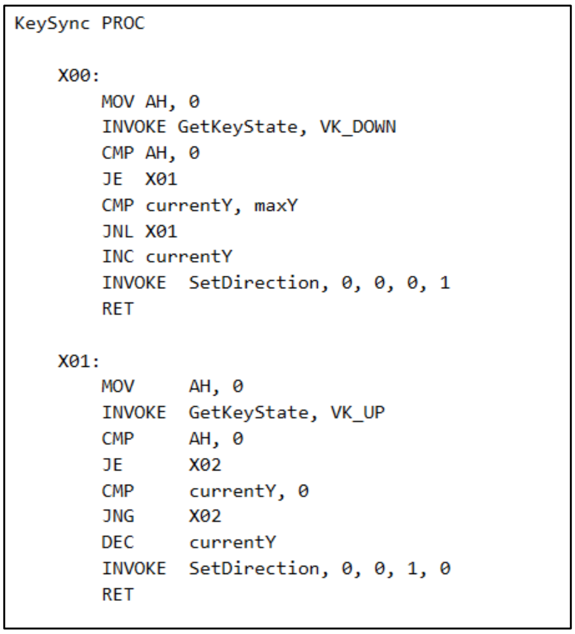
Description: To confirm the user action.



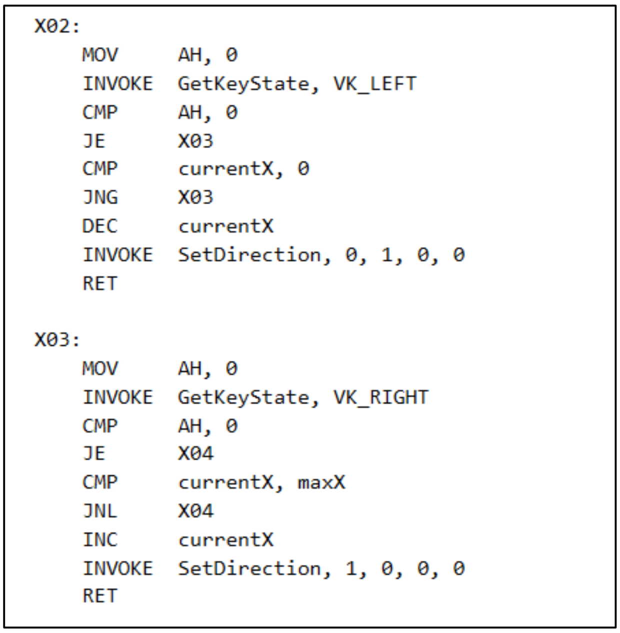
```
mGotoxy 30, 10
    mWrite  "Are you READY?"
    mGotoxy 30, 12
    mWrite  "0) YES, I AM READY FOR IT!"
    mGotoxy 30, 13
    mWrite  "1) No :("
    mGotoxy 30, 14
```

Module Key Sync:

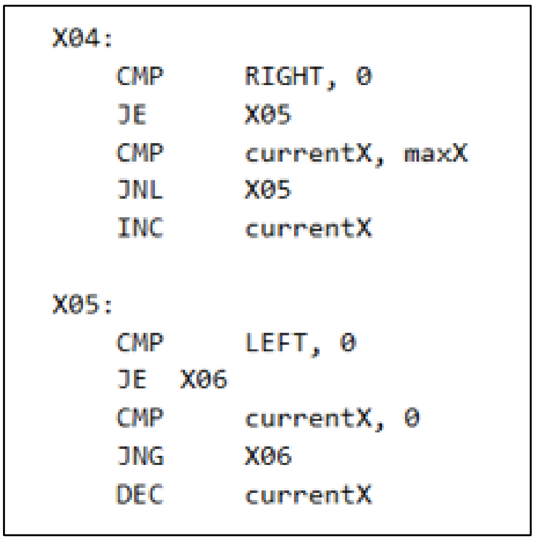
Description: To accept the input by user from keyboard, convert to processed data and move snake to destined direction.



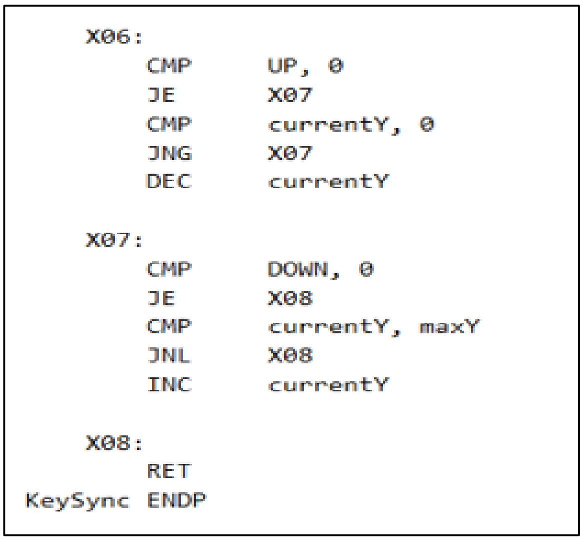
```
KeySync PROC

    X00:
        MOV AH, 0
        INVOKE GetKeyState, VK_DOWN
        CMP AH, 0
        JE  X01
        CMP currentY, maxY
        JNL X01
        INC currentY
        INVOKE  SetDirection, 0, 0, 0, 1
        RET

    X01:
        MOV     AH, 0
        INVOKE  GetKeyState, VK_UP
        CMP     AH, 0
        JE      X02
        CMP     currentY, 0
        JNG     X02
        DEC     currentY
        INVOKE  SetDirection, 0, 0, 1, 0
        RET
```



```
X02:
    MOV     AH, 0
    INVOKE  GetKeyState, VK_LEFT
    CMP     AH, 0
    JE      X03
    CMP     currentX, 0
    JNG     X03
    DEC     currentX
    INVOKE  SetDirection, 0, 1, 0, 0
    RET

X03:
    MOV     AH, 0
    INVOKE  GetKeyState, VK_RIGHT
    CMP     AH, 0
    JE      X04
    CMP     currentX, maxX
    JNL     X04
    INC     currentX
    INVOKE  SetDirection, 1, 0, 0, 0
    RET
```



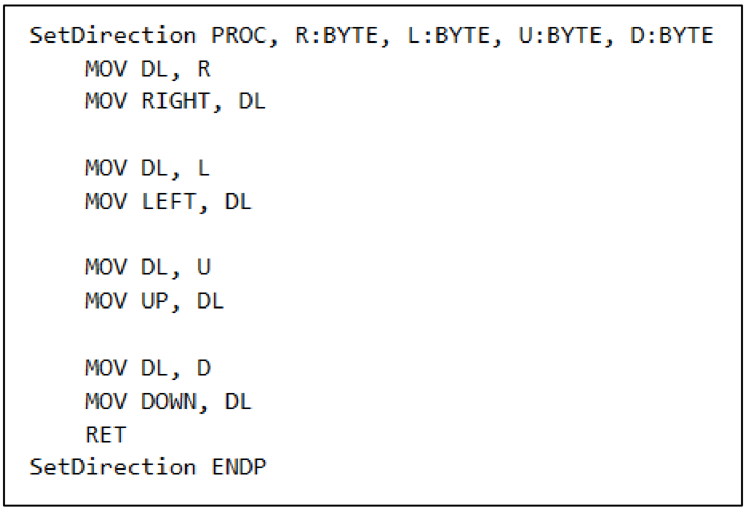
```
X04:
    CMP     RIGHT, 0
    JE      X05
    CMP     currentX, maxX
    JNL     X05
    INC     currentX

X05:
    CMP     LEFT, 0
    JE  X06
    CMP     currentX, 0
    JNG     X06
    DEC     currentX
```



```
    X06:
        CMP     UP, 0
        JE      X07
        CMP     currentY, 0
        JNG     X07
        DEC     currentY

    X07:
        CMP     DOWN, 0
        JE      X08
        CMP     currentY, maxY
        JNL     X08
        INC     currentY

    X08:
        RET
KeySync ENDP
```

Module Set Direction:

Description: To set the direction based on the processed input from user and return back to KeySync.

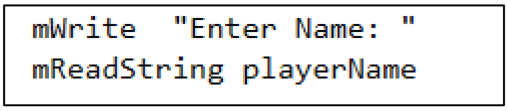


```
SetDirection PROC, R:BYTE, L:BYTE, U:BYTE, D:BYTE
    MOV DL, R
    MOV RIGHT, DL

    MOV DL, L
    MOV LEFT, DL

    MOV DL, U
    MOV UP, DL

    MOV DL, D
    MOV DOWN, DL
    RET
SetDirection ENDP
```

Liew Zu Xian

Module A:

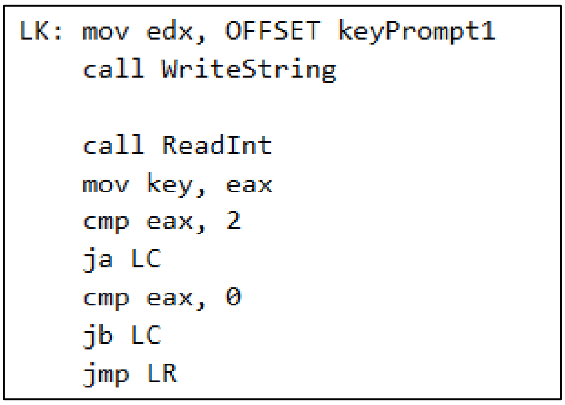
Description: To allow user to enter their name.



```
mWrite  "Enter Name: "
mReadString playerName
```

Module B:

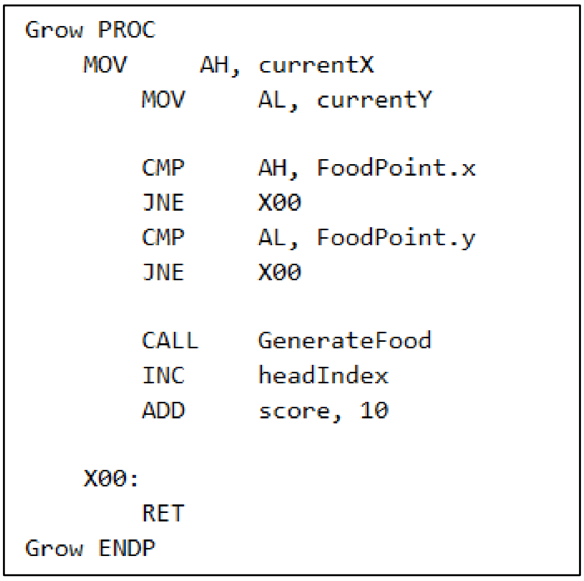
Description: To allow user to key data from keyboard into the project.



```
LK: mov edx, OFFSET keyPrompt1
    call WriteString

    call ReadInt
    mov key, eax
    cmp eax, 2
    ja LC
    cmp eax, 0
    jb LC
    jmp LR
```

Module Grow:

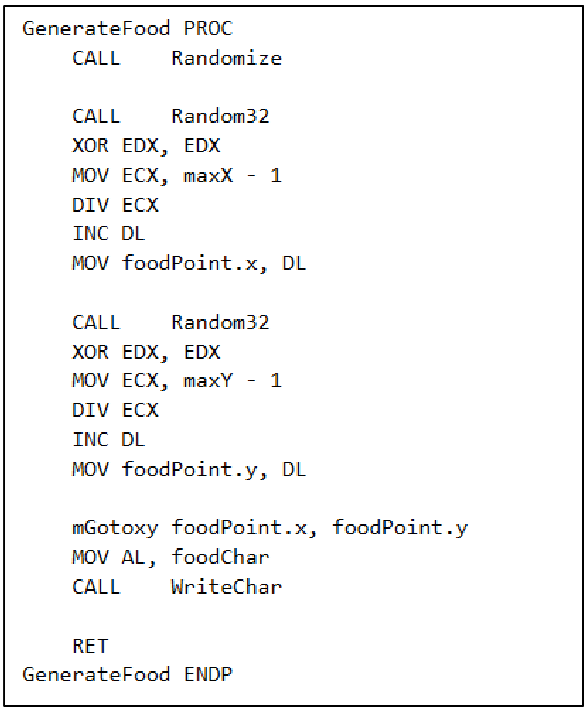
Description: To increase the length of the snake when the snake collided with food.



```
Grow PROC
    MOV     AH, currentX
        MOV     AL, currentY

        CMP     AH, FoodPoint.x
        JNE     X00
        CMP     AL, FoodPoint.y
        JNE     X00

        CALL    GenerateFood
        INC     headIndex
        ADD     score, 10

    X00:
        RET
Grow ENDP
```

Module Generate Food:

Description: To generate the food at random x,y coordinate for the snake.

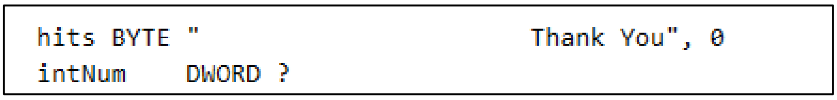


```
GenerateFood PROC
    CALL    Randomize

    CALL    Random32
    XOR EDX, EDX
    MOV ECX, maxX - 1
    DIV ECX
    INC DL
    MOV foodPoint.x, DL

    CALL    Random32
    XOR EDX, EDX
    MOV ECX, maxY - 1
    DIV ECX
    INC DL
    MOV foodPoint.y, DL

    mGotoxy foodPoint.x, foodPoint.y
    MOV AL, foodChar
    CALL    WriteChar

    RET
GenerateFood ENDP
```

William Liang Lenois

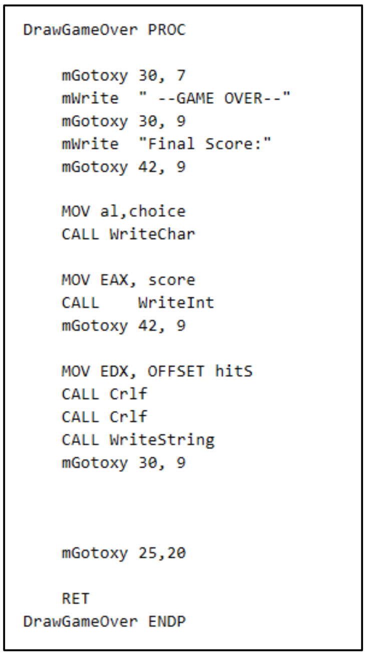
Module E:

Description: Display game over screen, final score and thank you note.



```
hits BYTE "                          Thank You", 0
intNum    DWORD ?
```

“Thank You” note initialized.

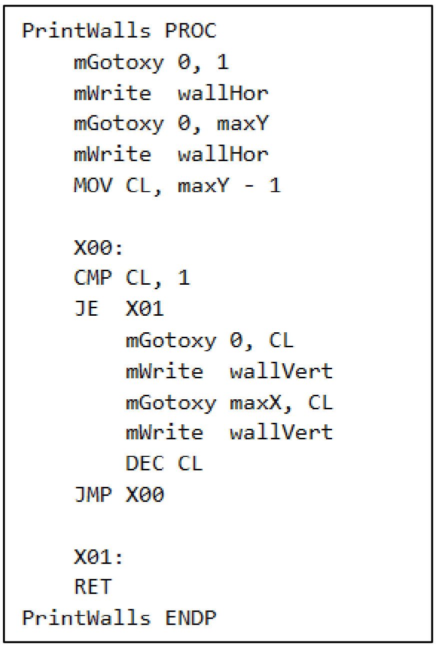


```
DrawGameOver PROC

    mGotoxy 30, 7
    mWrite  " --GAME OVER--"
    mGotoxy 30, 9
    mWrite  "Final Score:"
    mGotoxy 42, 9

    MOV al,choice
    CALL WriteChar

    MOV EAX, score
    CALL    WriteInt
    mGotoxy 42, 9

    MOV EDX, OFFSET hitS
    CALL Crlf
    CALL Crlf
    CALL WriteString
    mGotoxy 30, 9



    mGotoxy 25,20

    RET
DrawGameOver ENDP
```

Display summary details.

Module Print Walls:

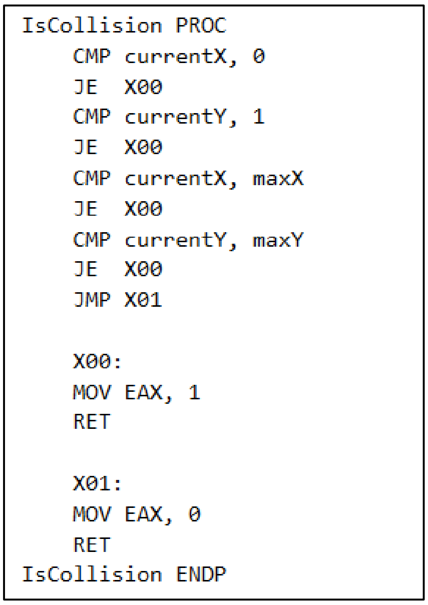
Description: To generate the wall for the snake.



```
PrintWalls PROC
    mGotoxy 0, 1
    mWrite  wallHor
    mGotoxy 0, maxY
    mWrite  wallHor
    MOV CL, maxY - 1

    X00:
    CMP CL, 1
    JE  X01
        mGotoxy 0, CL
        mWrite  wallVert
        mGotoxy maxX, CL
        mWrite  wallVert
        DEC CL
    JMP X00

    X01:
    RET
PrintWalls ENDP
```

Module Is Collision:

Description: To stop the game when the snake hits the wall.



```
IsCollision PROC
    CMP currentX, 0
    JE  X00
    CMP currentY, 1
    JE  X00
    CMP currentX, maxX
    JE  X00
    CMP currentY, maxY
    JE  X00
    JMP X01

    X00:
    MOV EAX, 1
    RET

    X01:
    MOV EAX, 0
    RET
IsCollision ENDP
```

Task Allocations:

Name	**Task Allocation**
Pua Kok Bin	Module D: Validation Module 1: Move Snake
Leong Zheng Jack	Module C: Confirmation Module 2: Key Sync Module 3: Set Direction
Liew Zu Xian	Module A: Login Module B: Test Data Module 4: Grow Module 5: Generate Food
William Liang Lenois	Module E: Completion Test/Rejection Test Module 6: Print Walls Module 7: Is Collision