Q#1 Nested Loop

```
section .data
    new_line db 0x0D, 0x0A
    asterisk db '*', 0

section .text
    global _start

_start:
    mov cx, 5           ; Outer loop counter (number of lines)

outer_loop:
    mov bx, 0           ; Inner loop counter (number of asterisks)

inner_loop:
    ; Print an asterisk
    mov ah, 0x0E
    mov al, [asterisk]
    int 0x10            ; BIOS interrupt to print character

    inc bx              ; Increment inner loop counter
    cmp bx, cx          ; Compare inner loop counter with outer loop counter
    jl inner_loop       ; If inner counter is less than outer counter, repeat

    ; Print a new line
    mov ah, 0x09
    lea dx, [new_line]
    int 0x21
```

```
    dec cx              ; Decrement outer loop counter
    jnz outer_loop      ; Repeat outer loop until counter is zero

exit:
    mov ah, 0x4C        ; Exit program
    int 0x21
```

Q#2 Procedures

```
section .data
    prompt_msg db 'Enter a number: $'
    result_msg db 'The square is: $'
    buffer db 50, 0
    new_line db 0x0D, 0x0A

section .bss
    result resb 4

section .text
    global _start

_start:
    mov ah, 0x09
    lea dx, [prompt_msg]
    int 0x21

    mov ah, 0x0A
    lea dx, [buffer]
    int 0x21
```

```
    mov al, [buffer + 1]   ; Get input number
    sub al, 48             ; Convert from ASCII to number

    call square            ; Call the square procedure

    mov ah, 0x09
    lea dx, [result_msg]
    int 0x21

    mov al, [result]       ; Load the result to print
    add al, 48             ; Convert back to ASCII
    mov ah, 0x0E
    int 0x10

    mov ah, 0x09
    lea dx, [new_line]
    int 0x21

exit:
    mov ah, 0x4C
    int 0x21

square:
    mov bl, al             ; Store the number in BL
    mul bl                 ; Multiply AL by BL (AL = AL * BL)
    mov [result], al       ; Store the result
    ret                    ; Return from procedure
```

Q#3 Macros

```
section .data
```

```
    prompt_msg db 'Enter a number (0-9): $'
    even_msg db 'The number is EVEN.$'
    odd_msg db 'The number is ODD.$'
    new_line db 0x0D, 0x0A

%macro PRINT_NUMBER 1
    mov ah, 0x0E
    add %1, 48          ; Convert number to ASCII
    mov al, %1
    int 0x10            ; Print character
%endmacro

section .text
    global _start

_start:
    mov ah, 0x09
    lea dx, [prompt_msg]
    int 0x21

    mov ah, 0x01
    int 0x21
    sub al, 48          ; Convert from ASCII to number

    ; Check if the number is even or odd
    and al, 1
    jz even

odd:
```

```
    mov ah, 0x09
    lea dx, [odd_msg]
    int 0x21
    PRINT_NUMBER 1      ; Print the number (odd case)
    jmp exit

even:
    mov ah, 0x09
    lea dx, [even_msg]
    int 0x21
    PRINT_NUMBER 0      ; Print the number (even case)

exit:
    mov ah, 0x4C
    int 0x21
```

Q#4 Create a procedure of your own choice

```
section .data
    prompt_msg db 'Enter a number (0-9): $'
    result_msg db 'The factorial is: $'
    new_line db 0x0D, 0x0A
    buffer db 50, 0

section .bss
    result resb 4

section .text
    global _start
```

```
_start:
    mov ah, 0x09
    lea dx, [prompt_msg]
    int 0x21

    mov ah, 0x0A
    lea dx, [buffer]
    int 0x21

    mov al, [buffer + 1]   ; Get input number
    sub al, 48             ; Convert from ASCII to number

    call factorial         ; Call the factorial procedure

    mov ah, 0x09
    lea dx, [result_msg]
    int 0x21

    mov al, [result]       ; Load the result to print
    add al, 48             ; Convert back to ASCII
    mov ah, 0x0E
    int 0x10

    mov ah, 0x09
    lea dx, [new_line]
    int 0x21

exit:
    mov ah, 0x4C
```

```
    int 0x21

factorial:
    cmp al, 0             ; Check if the number is 0
    je factorial_zero     ; If zero, factorial is 1
    push ax               ; Save the current value of AX
    dec al                ; Decrement the number
    call factorial        ; Recursive call
    pop bx                ; Retrieve the saved AX
    mov bl, al            ; Move result from AL to BL
    mov al, bx            ; Restore original value of AX
    mul bl                ; Multiply AL (n) by BL (factorial(n-1))
    mov [result], al      ; Store the result
    ret

factorial_zero:
    mov al, 1             ; Factorial of 0 is 1
    mov [result], al      ; Store the result
    ret
```