Q3. SOLUTION

```
;header as appropriate

SECTION         .bss
        m: dw
        n: dw
        t: dw

SECTION         .data
        value_min:    dw    0
        vslue_max:    dw    6
        factorial:    dw    1
        errorMsg:           db    "Number must be between 1 and 6 inclusive", 0
        retry:              db    "Please, try to enter a number again", 0

SECTION         .text
        main:

        START:
                call readInput
                cmp n, value_min
                je ERROR
                cmp n, value_max
                jg ERROR
                call computeFact                ;this is the key procedure
                call displayResult
                ret

        ERROR:
                call displayErrorMsg
                jmp START


        readInput:
                mov ah, 01h                     ;initialise standard input
                int 21h                         ;kernel command to store user
input
                mov n, al                       ;initialise n
                 ret


        displayResult:
                pop factorial
                mov dx, factorial
                mov ah, 09h
                int 21h
                ret


    displayErrorMsg:
                mov dx, [errorMsg]
                mov ah, 09h
```

```
        int 21h
        ret


computeFact:
    LOOP:
        mov m, n
        dec n
        mov t, ax
        cmp t, value_min
        je RESULT
        mult factorial, m
        mov factorial, ax
        mov ax, t
        mov n, ax
        jmp LOOP

    RESULT:
        push factorial
        ret
```

Q4.

```
    ;Header comments as appropriate

    SECTION     .bss
                s:      dw
                fGrade: db

    SECTION     .data
                min_score:  dw  0
                max_score:  dw  100
                score35:    dw  35
                score39:    dw  39
                score49:    dw  49
                score59:    dw  59
                score69:    dw  69
                score79:    dw  79
                score89:    dw  89
                gradeAp:    db  "A+", 0
                gradeA:     db  "A", 0
                gradeBp:    db  "B+", 0
                gradeB:     db  "B", 0
                gradeCp:    db  "C+", 0
                gradeC:     db  "C", 0
                gradeDp:    db  "D+", 0
                gradeD:     db  "D", 0
                errorMsg:   db  "The range of student's mark is 0 to 100", 0

    SECTION     .text
```

```
main:
        START:
                mov ax, 0
                call readUserInput
                pop s
                cmp s, min_score
                jl ERROR
                cmp s, max_score
                jg      ERROR
                call processGrade
                call displayFinalGrade
                ret

        ERROR:
                call displayErrorMsg
                jmp START


readUserInput:
                mov ah, 01h
                int 21h
                mov s, ah               ;initialise s from user input
                push s
                ret


displayErrorMsg:
                mov dx, [errorMsg]
                mov ah, 09h
                int 21h
                ret


displayFinalGrade:
                pop fGrade
                mov dx, [fGrade]
                mov ah, 09h
                int 21h
                ret


processGrade:
                pop s
                cmp s, score35
                jl      LD
                cmp s, score39
                jl      LDP
                cmp s, score49
                jl      LC
                cmp s, score59
                jl      LCP
                cmp s, score69
```

```
                jl      LB
                cmp s, score79
                jl      LBP
                cmp s, score89
                jl      LA
                jmp LAP

        LD:     mov ax, [gradeD]
                call setStudentGrade
                jmp FINISH

        LDP:    mov ax, [gradeDp]
                call setStudentGrade
                jmp FINISH

        LC:     mov ax, [gradeC]
                call setStudentGrade
                jmp FINISH

        LCP:    mov ax, [gradeCp]
                call setStudentGrade
                jmp FINISH

        LB:     mov ax, [gradeB]
                call setStudentGrade
                jmp FINISH

        LBP:    mov ax, [gradeBp]
                call setStudentGrade
                jmp FINISH

        LA:     mov ax, [gradeA]
                call setStudentGrade
                jmp FINISH

        LAP:    mov ax, [gradeAp]
                call setStudentGrade
                jmp FINISH

        FINISH: call resetRegisters
                ret


setStudentGrade:
                mov fGrade, ax
                push fGrade
                ret


resetRegisters:
                mov ax, min_score
                mov bx, min_score
```

```
mov cx, min_score
mov dx, min_score
mov ex, min_score
ret
```