Name – Mohiuddin Mondal
Roll – 001910501043
System programming
Assignment 1

Q1) Write and test a MASM program to Display your name and program title on the output screen.
Code:

```
 ;Write and test a MASM program to Display your name and program title on the output
screen.

 .MODEL SMALL
 .STACK 100H
 .DATA
    STRING_1  DB  'My name: Mohiuddin MOndal$'
    STRING_2  DB  'Assembly Programming$'

 .CODE
   MAIN PROC
     MOV AX, @DATA                ; initialize DS
     MOV DS, AX
     LEA DX, STRING_1             ; load & display the STRING_1
     MOV AH, 9
     INT 21H
     MOV AH, 2                    ; carriage return
     MOV DL, 0DH
     INT 21H
     MOV DL, 0AH                  ; line feed
     INT 21H
     LEA DX, STRING_2             ; load & display the STRING_2
     MOV AH, 9
     INT 21H
     MOV AH, 4CH                  ; return control to DOS
     INT 21H
   MAIN ENDP
 END MAIN
```

output:

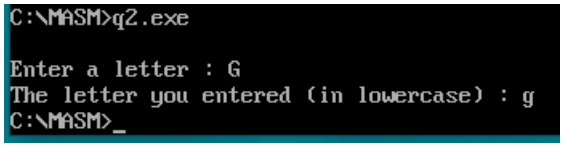
```
> ./q1
My name: Mohiuddin Mondal
Program title: Assembly Programming
```

Q2) Write and test a MASM program to convert a letter from uppercase to lowercase.
Code:

```
.MODEL SMALL
 .STACK 100H
 .DATA
    INPUT db 13,10, "Enter a letter : $"
    OUTPUT db  13, 10, "The letter you entered (in lowercase) : $"

 .CODE
      MAIN PROC
             MOV AX, @DATA;initialize DS
             MOV DS, AX
             LEA DX, INPUT;show the input prompt
             MOV AH, 9
             INT 21H
             MOV AH, 1           ;read letter
             INT 21H
             LEA DX, OUTPUT      ;show output prompt
             MOV AH, 9
             INT 21H
             MOV BL, AL          ;store the letter in BL
             CMP BL, 'A'         ;if the character is <A goto loop1
             JL LOOP1
             CMP BL, 'Z'         ;if the character is >Z goto loop1
             JG LOOP1
             ADD BL, 20H         ;else convert into lowercase
```

```
      LOOP1: MOV AH, 2          ;write the output letter
             MOV DL, BL
             INT 21H
             MOV AH, 4CH        ;return control to DOS
             INT 21H
       MAIN ENDP
END MAIN
```

output:



Q3) Write and test a MASM program to add two Hexadecimal Numbers.

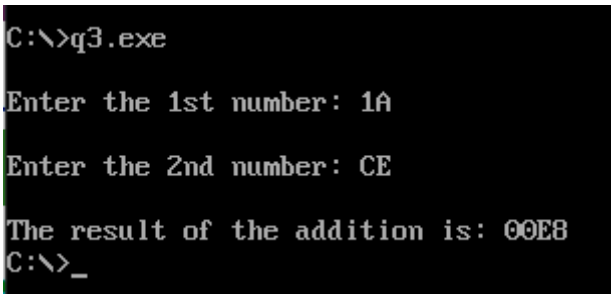
Code:

```
.MODEL SMALL
.STACK 100H
.DATA
    PROMPT1 db 13,10,"Enter the 1st number: $"
    PROMPT2 db 13,10,"Enter the 2nd number: $"
    PROMPT3 db 13,10,"The result of the addition is: $"

.CODE
   MAIN PROC
          MOV AX,@DATA                                ;for moving data to data segment
          MOV DS,AX
          XOR BX,BX                                   ;initially BX value is equal to 0
          MOV CL,4
          LEA DX, PROMPT1                             ;show num1 prompt
          MOV AH, 9
          INT 21H
          MOV AH,1                                    ;for taking input
          INT 21h

      INPUT1:
          CMP AL,0DH                 ;compare whether the pressed key is 'ENTER' or not
          JE LINE1               ;If it is equal to 'Enter' then stop taking first value

          CMP AL,39h             ;compare the input whether it is letter or digit.39h is
                                 the ascii value of 9
          JG LETTER1

          AND AL,0FH              ;if it is digit then convert it's ascii value to real
                                   value by masking


      LETTER1:                    ;if it is letter then subtract 37h from it to find it's
                                        real value
          SUB AL,37h

      SHIFT1:
          SHL BX, CL
          OR  BL,AL                                   ;making 'or' will add the current
                                                           value with previous value

          INT 21h
          JMP INPUT1
      LINE1:
          LEA DX, PROMPT2                             ;show num2 prompt
          MOV AH, 9
          INT 21H

          XOR DX,DX                                   ;set dx value zero

          MOV AH,1
          INT 21h


      INPUT2:
```

```
        CMP AL,0DH                          ;compare whether the pressed key is
                                                 'ENTER' or not
        JE LINE2                            ;If it is equal to 'Enter' then
                                                 stop taking first value

        CMP AL,39h                          ;compare the input whether it is
                                           letter or digit.
        JG LETTER2

        AND AL,0FH                          ;if digit then convert it's
                                           ascii value to real value by masking
        JMP SHIFT2

    LETTER2:                                ;if it is letter then subtract 37h
                                           from it to find it's real value
        SUB AL,37H

    SHIFT2:
        SHL DX, CL
        OR  DL,AL                           ;making 'or' will add the current
                                                 value with previous value

        INT 21h
        JMP INPUT2
    LINE2:
        XOR CX,CX
        MOV CX,DX

        MOV DH,16

    SUM:
        ADD BX,CX                          ;add two number which are stored in
bx and cs register
        JC PC1                             ;if the register is overflowed then
                                                 print an extra 1

        mov cl, 4

        LEA DX, PROMPT3                    ;show answer prompt
        MOV AH, 9
        INT 21H

        OUTPUT:                            ;level for printing their sum

        MOV CH,BH
        SHR CH, CL
        AND CH,0FH

        CMP CH,10                          ;convert decimal to binary
        add cH,'0'
        cmp cH,':'
        jL TAG
        add ch,7
    TAG:MOV DL,CH
        MOV AH,2
        INT 21h

        MOV CH,BH
        AND CH,0FH
        CMP CH,10
        add cH,'0'
        cmp cH,':'
        jL TAG1
        add ch,7
    TAG1:MOV DL,CH
        MOV AH,2
        INT 21h

        MOV CH,BL
        SHR CH, CL
        AND CH,0FH
        CMP CH,10
        add cH,'0'
        cmp cH,':'
        jL TAG2
        add ch,7
    TAG2:MOV DL,CH
        MOV AH,2
        INT 21h
```

```
        MOV CH,BL
        AND CH,0FH
        CMP CH,10
        add cH,'0'
        cmp cH,':'
        jL TAG3
        add ch,7
    TAG3:MOV DL,CH
        MOV AH,2
        INT 21h

        JMP EXIT
    PC1:                                    ;level for printing overflowed 1
        MOV DL,'1'
        MOV AH,2
        INT 21h
        JMP OUTPUT

    EXIT:
        MOV AH, 4CH                         ;return control to DOS
        INT 21H

  MAIN ENDP
 END MAIN
```

Output:



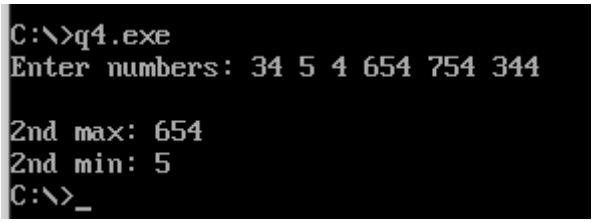
Q4)Write and test a MASM program to find the second max and second min from an array.

Code:

```
        .model small
        .stack 256
CR equ 13d
LF equ 10d
        .data
prompt1 db 'Enter numbers: $'
max2msg db CR, LF,'2nd max: $'
min2msg db CR, LF, '2nd min: $'
max1 dw ?
max2 dw ?
min1 dw ?
min2 dw ?
        .code
start:
        mov ax, @data
        mov ds, ax
        mov max1,0
        mov max2,0
        mov min1,9999
        mov min2,9999
        mov dx, offset prompt1
        mov ah,9
        int 21h
        l0:  mov bx,0
        l1:  mov ah,1
        int 21h
        cmp al,13
        je l3
        cmp al,32
```

```
        je l2
        sub al,'0'
        push ax
        mov ax,bx
        mov cx,10
        mul cx
        mov bx,ax
        pop ax
        mov ah,0
        add bx,ax
        jmp l1
        l2:  cmp max1,bx
        jnl l4
        mov cx,max1
        mov max2,cx
        mov max1,bx
        jmp lb2
        l4:  cmp max2,bx
        jnl lb2
        mov max2,bx
        jmp lb2
        lb2:  cmp min1,bx
        jng lb4
        mov cx,min1
        mov min2,cx
        mov min1,bx
        jmp l0
        lb4:  cmp min2,bx
        jng l0
        mov min2,bx
        jmp l0
        l3:  mov dx, offset max2msg
        mov ah,9
        int 21h
        mov ax,max2
        call putn
        mov dx, offset min2msg
        mov ah,9
        int 21h
        mov ax,min2
        call putn
        mov ax, 4c00h
        int 21h
putn:           ; display number in ax
                ; ax contains number (and also div C in above)
                ; dx contains remainder (rem in C above)
                ; cx contains 10 for division
    push    bx
    push    cx
    push    dx

    mov     dx, 0           ; dx = 0
    push    dx          ; push 0 as sentinel
    mov     cx, 10      ; cx = 10

    cmp     ax, 0
    jge     calc_digits     ; number is negative
    neg     ax          ; ax = -ax; ax is now positive
    push    ax          ; save ax
    mov al, '-'     ; display - sign
    call    putc
    pop ax          ; restore ax

calc_digits:
    div cx          ; dx:ax = ax / cx
                    ; ax = result, dx = remainder
    add dx, '0'     ; convert dx to digit
    cmp dx,57
    jle l33
    add dx,7
    l33:            push    dx          ; save digit on stack
    mov dx, 0       ; dx = 0
    cmp ax, 0           ; finished ?
    jne calc_digits     ; no, repeat process

;   all digits now on stack, display them in reverse

disp_loop:
    pop dx          ; get last digit from stack
    cmp dx, 0           ; is it sentinel
```

```
    je  end_disp_loop   ; if yes, we are finished
    mov al, dl      ; al = dl
    call    putc            ; otherwise display digit
    jmp disp_loop

end_disp_loop:
    pop dx          ; restore registers
    pop cx
    pop bx
    ret

putc: ; display character in al

        push ax ; save ax
        push bx ; save bx
        push cx ; save cx
        push dx ; save dx

        mov dl, al
        mov ah, 2h
        int 21h

        pop dx ; restore dx
        pop cx ; restore cx
        pop bx ; restore bx
        pop ax ; restore ax
        ret
end start
```



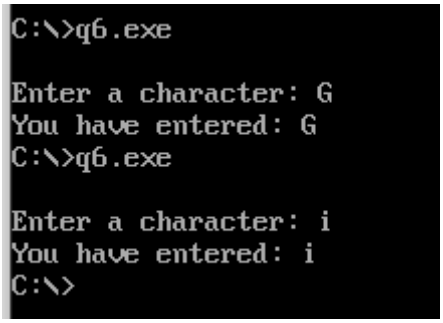
Q5) Write and test a MASM program to display a terminating message.

Code:

```
.MODEL SMALL
.STACK 100H

.DATA

  TERM_PROMPT DB '################# My terminate message ###################$'

.CODE
  MAIN PROC
    MOV AX,@DATA
    MOV DS,AX

    LEA DX, TERM_PROMPT
    MOV AH,9
    INT 21H

    MOV AH,4CH
    INT 21H
  MAIN ENDP
END MAIN
```

Q6) Write and test a MASM program to Take a character from the keyboard and print it.
Code:

```
; Write and test a MASM program to Take a character from keyboard and print it.
.model small
.stack 100h

.data
msg1 db 10,13,"Enter a character: $"
msg2 db 10,13,"You have entered: $"

.code

main proc

       mov ax,@data
       mov ds,ax

       ;display input prompt
       lea dx,msg1
       mov ah,09h
       int 21h

       ;accept a character
       mov ah,01h
       int 21h

       ;al has the character
       ;display prompt
       lea dx,msg2
       mov ah,09h
       int 21h

       ;display the character
       mov dl,al
       mov ah,02h
       int 21h

       mov ah,4ch
       int 21h

main endp
end main
```



Q7) Write and test a MASM program to validate second numbers is less than the first.

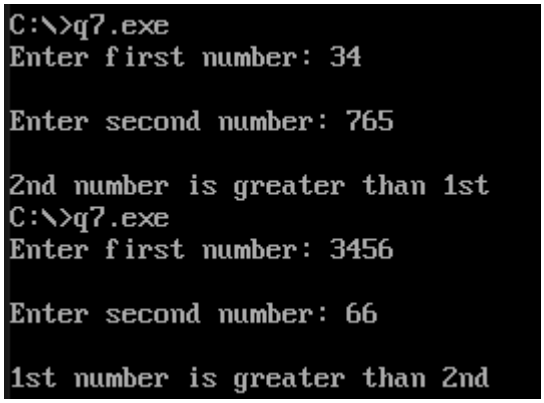
Code:

```
        .model small
        .stack 256

CR equ 13d
LF equ 10d

        .data
```

```
prompt1 db 'Enter first number: $'
prompt2 db CR, LF,'Enter second number: $'
output1 db CR, LF, 'Both are Equal numbers$'
output2 db CR, LF, '2nd number is greater than 1st$'
output3 db CR, LF, '1st number is greater than 2nd$'
num1 dw ?
num2 dw ?

        .code
start:
        mov ax, @data
        mov ds, ax

        mov ax, offset prompt1
        call puts ; display prompt1

        call getn ; read first number
        mov num1, ax
        mov ax, offset prompt2
        call puts ; display prompt2

        call getn ; read second number

        mov num2, ax

        mov ax, num1 ; ax = num1
        mov bx, num2 ; bx = num2
        cmp ax,bx
  je ll1
  jl ll2
  mov ax, offset output3
        call puts
  jmp ll
  ll2:  mov ax, offset output2
        call puts
  jmp ll
  ll1:  mov ax, offset output1
        call puts
        ll: mov ax, 4c00h
        int 21h ; finished, back to dos

getn:           ; read a number from the keyboard
            ; return value in ax register
;                                       C variables
            ; dx records sign of number         sign variable
            ; bl stores each digit              digit variable
            ; cx stores the number read in so far   n variable
            ; al stores each character read in.     c variable
            ; ax is also used in the mul instruction


        push bx         ; save registers on stack
        push cx
        push dx

        mov dx, 1       ; record sign, 1 for positive
        mov bx, 0       ; initialise digit to 0
        mov cx, 0       ; initialise number to 0

        call getc       ; read first character
        cmp al, '-'     ; is it negative
        jne newline     ; if not goto newline
        mov dx, -1      ; else record sign

        call getc       ; get next digit
newline:
        push dx     ; save sign on stack
        cmp al, 13      ; (al == CR) ?
        je  fin_read    ; if yes, goto fin_read
                    ; otherwise
        sub al, '0'     ; convert to digit
        cmp al,9
        jle l1
        sub al,7
        l1: mov cl, al      ; cl = first digit
        call getc       ; get next character

read_loop:
        cmp al, 13      ; if (al == CR)
        je fin_read     ; then goto fin_read
```

```
         sub al, '0'    ; otherwise, convert to digit
         cmp al,9
         jle l2
         sub al,7
         l2:  mov bl, al     ; bl = digit
         mov ax, 16     ; ax = 10
         mul cx     ; ax = cx * 10
         mov cx, ax     ; cx = ax  n = n * 10
         add cx, bx     ; cx = cx + digit  n = n + digit
         call getc      ; read next digit
        jmp read_loop

fin_read:
        mov ax, cx      ; number returned in ax
        pop dx      ; retrieve sign from stack
        cmp dx, 1       ; ax = ax * dx
        je  fin_getn
        neg ax      ; ax = -ax
fin_getn:
        pop dx
        pop cx
        pop bx
        ret

puts:       ; display a string terminated by $
            ; dx contains address of string

        push ax ; save ax
        push bx ; save bx
        push cx ; save cx
        push dx ; save dx

        mov dx, ax
        mov ah, 9h
        int 21h ; call ms-dos to output string

        pop dx ; restore dx
        pop cx ; restore cx
        pop bx ; restore bx
        pop ax ; restore ax

        ret

putn:               ; display number in ax
                ; ax contains number (and also div C in above)
                ; dx contains remainder (rem in C above)
                ; cx contains 10 for division
    push    bx
    push    cx
    push    dx

    mov     dx, 0           ; dx = 0
    push    dx          ; push 0 as sentinel
    mov     cx, 16      ; cx = 10

    cmp     ax, 0
    jge     calc_digits     ; number is negative
    neg     ax          ; ax = -ax; ax is now positive
    push    ax          ; save ax
    mov al, '-'     ; display - sign
    call    putc
    pop ax          ; restore ax

calc_digits:
    div cx          ; dx:ax = ax / cx
                    ; ax = result, dx = remainder
    add dx, '0'     ; convert dx to digit
    cmp dx,57
    jle l3
    add dx,7
    l3:   push    dx          ; save digit on stack
    mov dx, 0       ; dx = 0
    cmp ax, 0           ; finished ?
    jne calc_digits     ; no, repeat process

;   all digits now on stack, display them in reverse

disp_loop:
    pop dx          ; get last digit from stack
```

```
    cmp dx, 0           ; is it sentinel
    je  end_disp_loop   ; if yes, we are finished
    mov al, dl      ; al = dl
    call    putc            ; otherwise display digit
    jmp disp_loop

end_disp_loop:
    pop dx          ; restore registers
    pop cx
    pop bx
    ret

putc: ; display character in al

        push ax ; save ax
        push bx ; save bx
        push cx ; save cx
        push dx ; save dx

        mov dl, al
        mov ah, 2h
        int 21h

        pop dx ; restore dx
        pop cx ; restore cx
        pop bx ; restore bx
        pop ax ; restore ax
        ret


getc:   ; read character into al

        push bx ; save bx
        push cx ; save cx
        push dx ; save dx

        mov ah, 1h
        int 21h

        pop dx ; restore dx
        pop cx ; restore cx
        pop bx ; restore bx
        ret

        end start
```



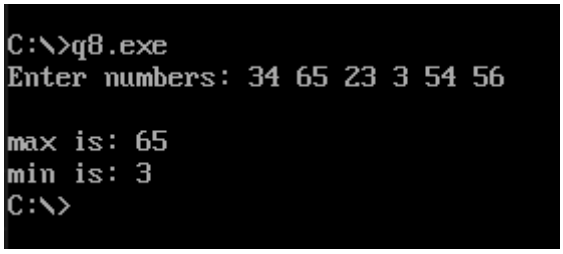
Q8) Write andtest a MASM program to find maximum and minimum from an array.

Code:

```
        .model small
        .stack 256
CR equ 13d
LF equ 10d
        .data
```

```
prompt1 db 'Enter numbers: $'
maxOutput db CR, LF,'max is: $'
minOutput db CR, LF, 'min is: $'
max1 dw ?
min1 dw ?
        .code
start:
        mov ax, @data
        mov ds, ax
        mov max1,0
        mov min1,9999
        mov dx, offset prompt1
        mov ah,9
        int 21h
        l0:  mov bx,0
        l1:  mov ah,1
        int 21h
        cmp al,13
        je l3
        cmp al,32
        je l2
        sub al,'0'
        push ax
        mov ax,bx
        mov cx,10
        mul cx
        mov bx,ax
        pop ax
        mov ah,0
        add bx,ax
        jmp l1
        l2:  cmp max1,bx
        jnl lb2
        mov max1,bx
        jmp lb2
        lb2:  cmp min1,bx
        jng l0
        mov min1,bx
        jmp l0

        l3:  mov dx, offset maxOutput
        mov ah,9
        int 21h
        mov ax,max1
        call putn
        mov dx, offset minOutput
        mov ah,9
        int 21h
        mov ax,min1
        call putn
        mov ax, 4c00h
        int 21h
putn:                ; display number in ax
                ; ax contains number (and also div C in above)
                ; dx contains remainder (rem in C above)
                ; cx contains 10 for division
    push    bx
    push    cx
    push    dx

    mov     dx, 0           ; dx = 0
    push    dx          ; push 0 as sentinel
    mov     cx, 10      ; cx = 10

    cmp     ax, 0
    jge     calc_digits     ; number is negative
    neg     ax          ; ax = -ax; ax is now positive
    push    ax          ; save ax
    mov al, '-'     ; display - sign
    call    putc
    pop ax          ; restore ax

calc_digits:
    div cx          ; dx:ax = ax / cx
                    ; ax = result, dx = remainder
    add dx, '0'     ; convert dx to digit
    cmp dx,57
    jle l33
    add dx,7
    l33:            push    dx          ; save digit on stack
```

```
    mov dx, 0       ; dx = 0
    cmp ax, 0           ; finished ?
    jne calc_digits     ; no, repeat process

;   all digits now on stack, display them in reverse

disp_loop:
    pop dx          ; get last digit from stack
    cmp dx, 0           ; is it sentinel
    je  end_disp_loop   ; if yes, we are finished
    mov al, dl      ; al = dl
    call    putc            ; otherwise display digit
    jmp disp_loop

end_disp_loop:
    pop dx          ; restore registers
    pop cx
    pop bx
    ret

putc: ; display character in al

        push ax ; save ax
        push bx ; save bx
        push cx ; save cx
        push dx ; save dx

        mov dl, al
        mov ah, 2h
        int 21h

        pop dx ; restore dx
        pop cx ; restore cx
        pop bx ; restore bx
        pop ax ; restore ax
        ret
end start
```

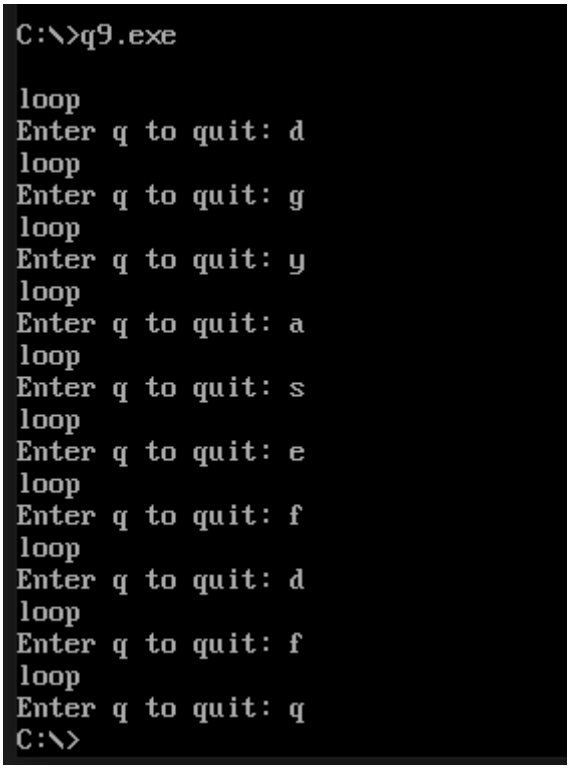


q9) Write and test a MASM program to loopuntil the user decides to quit.

Code:

```
.model small
.stack 100h

.data
msg db 10,13,"Enter q to quit: $"
looping db 10,13,"loop$"

.code

main proc

       mov ax,@data
       mov ds,ax

       label1:

              ;display loop message
              lea dx,looping
              mov ah,09h
```

```
            int 21h

            ;display input prompt
            lea dx,msg
            mov ah,09h
            int 21h

            ;accept a character
            mov ah,01h
            int 21h

            ; check if character is q
            cmp al,'q'
            jne label1

       ;exit
       mov ah,4Ch
       int 21h
main endp
end main
```



Q10) Write and test a MASM program to print all the characters from A-Z.

Code:

```
; Write and test a MASM program to Print all the characters from A-Z.
.model small
.stack 100h

.data
space db ' '
.code

main proc

       mov ax,@data
       mov ds,ax

       mov bx,65
       mov cx,0

       label1:
```

```
            ;pr int the character
            mov ah,02h
            mov dl,bl
            int 21h

            ;print the character
            mov ah,02h
            mov dl,space
            int 21h

            ;increment
            inc bx
            inc cx
            cmp cx,26

      jne label1

      mov ah,4ch
      int 21h

main endp

end main
```