Task 01:

```
; Q1 solution starts from here

.MODEL small
.STACK 100h
.DATA
   msg1 DB 'Enter the length of your name: $'
   msg2 DB 'Enter your name: $'
   msg3 DB 'Your name is: $'
   buffer DB 50 DUP('$')
   length DB ?


.CODE
main PROC
; Initialize data segment
MOV AX, @DATA
MOV DS, AX

;Length
LEA DX, msg1
MOV AH, 09h
INT 21h

;Input
MOV AH, 01h
INT 21h
SUB AL, '0'
MOV length, AL

MOV AH, 01h
INT 21h
```

```
;FOR name
LEA DX, msg2
MOV AH, 09h
INT 21h

;Input name
LEA DX, buffer
MOV AH, 0Ah
INT 21h

;Display name
LEA DX, msg3
MOV AH, 09h
INT 21h

LEA DX, buffer+2
MOV AH, 09h
INT 21h

;Terminate program
   MOV AH, 4Ch
   INT 21h
main ENDP
END main



; Q1 solution ends here
```

Task 02:

```
; Q2 solution starts from here
.MODEL small
.STACK 100h
.DATA
   msg1 DB 'Enter 5 numbers: $'
   msg2 DB 'Sorted array: $'
   array DB 5 DUP(?)
```

```
.CODE
main PROC
;Initialize data segment
MOV AX, @DATA
MOV DS, AX


LEA DX, msg1
MOV AH, 09h
INT 21h

;5numbers
MOV CX, 5
LEA DI, array

read_numbers:
MOV AH, 01h
INT 21h
SUB AL, '0'
MOV [DI],AL
INC DI
LOOP read_numbers


MOV CX, 4


outer_loop:
MOV SI, 0
MOV DI, SI

inner_loop:
MOV AL, [array + SI] ;First element
CMP AL, [array + SI + 1]
JBE no_swap
;SWAP
MOV BL, AL
MOV AL, [array + SI + 1]
MOV [array + SI], AL
MOV AL, BL
MOV [array + SI + 1], AL

no_swap:
INC SI
```

```
CMP SI, CX
JL inner_loop

LOOP outer_loop

;Printing sorted array
LEA DX, msg2
MOV AH, 09h
INT 21h

MOV CX, 5
LEA DI, array

print_numbers:
        MOV AL, [DI]
        ADD AL, '0'
        MOV DL, AL
        MOV AH, 02h
        INT 21h
        MOV DL, ' '
        MOV AH, 02h
        INT 21h
        INC DI
        LOOP print_numbers

;Terminate program
    MOV AH, 4Ch
    INT 21h
main ENDP
END main
```

; Q2 solution ends here

Task 03:

```
; Q3 solution starts from here
.MODEL small
.STACK 100h
```

```
.DATA
    msg1 DB 'Enter three numbers: $'
    msg2 DB 'The maximum number is: $'
    num1 DB ?
    num2 DB ?
    num3 DB ?
    max DB ?

.CODE
main PROC
    ; Initialize data segment
    MOV AX, @DATA
    MOV DS, AX

    ; Input numbers
    LEA DX, msg1
    MOV AH, 09h
    INT 21h

    ; Read three numbers
    MOV CX, 3
    MOV DI, OFFSET num1

    read_numbers:
        MOV AH, 01h
        INT 21h
        SUB AL, '0'
        MOV [DI], AL
        INC DI
        LOOP read_numbers

    ; Find maximum
    MOV AL, num1
    MOV max, AL

    MOV DI, OFFSET num2

    find_max:
        MOV BL, [DI]
        CMP AL, BL
        JGE next_num
        MOV AL, BL
        MOV max, AL
```

```
    next_num:
        INC DI
        CMP DI, OFFSET num3 + 1
        JL find_max

    ;Print
    LEA DX, msg2
    MOV AH, 09h
    INT 21h

    MOV DL, max
    ADD DL, '0'
    MOV AH, 02h
    INT 21h

    ; Terminate program
    MOV AH, 4Ch
    INT 21h
main ENDP
END main

; Q3 solution ends here
```

Task 04:

Q4:

```
.MODEL small
.STACK 100h
.DATA
    msg1 DB 'Enter a number: $'
    msg2 DB 'Factors of the number: $'
    buffer DB 4 DUP('$')
    num DW ?

.CODE
main PROC
    ; Initialize data segment
```

```
MOV AX, @DATA
MOV DS, AX
MOV SS, AX
MOV SP, OFFSET buffer
LEA DX, msg1
MOV AH, 09h
INT 21h

LEA DX, buffer
MOV AH, 0Ah
INT 21h

MOV SI, offset buffer + 2
MOV AL, [SI]
SUB AL, '0'
MOV BL, AL
MOV AL, [SI+1]
SUB AL, '0'
MOV AH, AL
MOV AL, BL
MOV BL, 10
MUL BL
ADD AL, AH
MOV num, AX

;push onto stack
MOV CX, num
MOV AL, 1

find_factors:
   MOV AX, num
   MOV BL, AL
   MOV DX, 0
   DIV BL
   CMP DX, 0
   JNE not_a_factor

   MOV AH, 0
   MOV AL, BL
   PUSH AX

not_a_factor:
   INC AL
   CMP AL, CL
```

```
        JLE find_factors


    LEA DX, msg2
    MOV AH, 09h
    INT 21h

    ; Display factors
    MOV CX, 0
    display_factors:
        POP AX
        MOV DL, AL
        ADD DL, '0'
        MOV AH, 02h
        INT 21h
        MOV DL, ' '
        MOV AH, 02h
        INT 21h
        INC CX
        CMP CX, num
        JBE display_factors

; Terminate program
    MOV AH, 4Ch
    INT 21h
main ENDP
END main
```

Task 05:

Q5

```
.MODEL small
.STACK 100h
.DATA
    msg1 DB 'Enter a string: $'
    msg2 DB 'Reversed string: $'
    buffer DB 50 DUP('$')
    length DB ?
```

```
.CODE
main PROC
; Initialize data segment
MOV AX, @DATA
MOV DS, AX


LEA DX, msg1
MOV AH, 09h
INT 21h


LEA DX, buffer
MOV AH, 0Ah
INT 21h


MOV AL, [buffer+1]
MOV length, AL


LEA SI, buffer+2
MOV CL, length
MOV CH, 0
MOV SP, 1000h ;stackpointer

push_loop:
    MOV AL, [SI]
    PUSH AX
    INC SI
    LOOP push_loop

;reverse string
LEA DX, msg2
MOV AH, 09h
INT 21h

MOV CX, 0 ; Clear CX
MOV CL, length ; Move the value of length to the 8-bit register CL
MOV SI, 1000h

pop_loop:
POP AX
MOV DL, AL
```

```
MOV AH, 02h
INT 21h

LOOP pop_loop

;Terminate program
   MOV AH, 4Ch
   INT 21h
main ENDP
END main
```

Task 06:

Q6:

```
.MODEL small
.STACK 100h
.DATA
   msg1 DB 'Enter a number: $'
   msg2 DB 'All digits are unique. $'
   msg3 DB 'Digits are not unique. $'
   num DB 6 DUP(?)
   length DB ?

.CODE
main PROC
; Initialize data segment
MOV AX, @DATA
MOV DS, AX

; Input number
LEA DX, msg1
MOV AH, 09h
INT 21h

MOV AH, 0Ah
LEA DX, num
INT 21h


MOV AH, 01h
INT 21h
```

```
MOV AL, [num+1]
MOV length, AL


MOV SI, 2
MOV CL, length
XOR CH, CH

  check_digits:
      MOV AL, [num+SI-1]
      SUB AL, '0'
      MOV DI, 2

     check_loop:
         CMP DI, SI
         JGE skip_loop
         MOV DL, [num+DI-1]
         SUB DL, '0'
         CMP AL, DL
         JE not_unique
         INC DI
         LOOP check_loop

     skip_loop:
     INC SI
     LOOP check_digits

   ;unique
   LEA DX, msg2
   MOV AH, 09h
   INT 21h

; Terminate program
   MOV AH, 4Ch
   INT 21h

   not_unique:
      LEA DX, msg3
      MOV AH, 09h
      INT 21h

   MOV AH, 4Ch
```

```
    INT 21h
main ENDP
END main
```

Task 07:

Q7:

```
.MODEL small
.STACK 100h
.DATA
    msg1 DB 'Enter a number: $'
    msg2 DB 'Prime numbers: $'
    buffer DB 2 DUP('$')
    num DB ?
    primes DB 101 DUP(0)
    i DB ?
    j DB ?

.CODE
main PROC
    ; Initialize data segment
    MOV AX, @DATA
    MOV DS, AX

    ; Input number
    LEA DX, msg1
    MOV AH, 09h
    INT 21h

    MOV AH, 01h
    INT 21h
    SUB AL, '0'
    MOV num, AL

    MOV CL, num
    MOV CH, 00h
    LEA DI, primes
```

```
    initialize_array:
        MOV [DI], 1
        INC DI
        LOOP initialize_array

    ; 0 and 1 r notprimes
    MOV BYTE PTR [primes], 0
    MOV BYTE PTR [primes + 1], 0
    MOV AL, 2
    MOV BL, num

sieve_loop:
    MOV DL, AL
    MOV SI, 0
    MOV CX, AX

    mark_multiples:
        MOV [primes + SI], 0
        ADD SI, CX
        CMP SI, BX
        JBE mark_multiples

    INC AL
    CMP AL, BL
    JBE sieve_loop



LEA DX, msg2
MOV AH, 09h
INT 21h

MOV CL, 2
LEA DI, primes
MOV BL, num

print_primes:
    MOV AL, [DI + CL - 2] CL - 2
    CMP AL, 1
    JNE not_prime

    MOV AL, CL
    ADD AL, '0'
    MOV AH, 02h
```

```
    INT 21h
    MOV AL, ' '
    MOV AH, 02h
    INT 21h

not_prime:
    INC CL
    CMP CL, BL
    JBE print_primes

    ; Terminate program
    MOV AH, 4Ch
    INT 21h
main ENDP
END main
```

Task 08:

Q8:

```
.MODEL small
.STACK 100h
.DATA
    msg1 DB 'Enter text: $'
    msg2 DB 'Reversed text: $'
    buffer DB 100 DUP('$')
    rev_buffer DB 100 DUP('$')
    space DB ' '
    newline DB 0Dh, 0Ah, '$'

.CODE
main PROC
    ; Initialize data segment
    MOV AX, @DATA
    MOV DS, AX

    LEA DX, msg1
```

```
MOV AH, 09h
INT 21h
LEA DX, buffer
MOV AH, 0Ah
INT 21h

;pointers
LEA SI, buffer + 2
LEA DI, rev_buffer
MOV CX, 0

process_text:
   MOV AL, [SI]
   CMP AL, ' '
   JE reverse_word
   CMP AL, 0Dh
   JE finish
   MOV [DI], AL
   INC DI
   INC SI
   JMP process_text

reverse_word:
   DEC DI
   reverse_loop:
      MOV AL, [DI]
      CMP AL, ' '
      JE append_space
      MOV BX, CX
      MOV [rev_buffer + BX], AL
      INC CX
      DEC DI
      JMP reverse_loop

   append_space:
      MOV BX, CX
      MOV [rev_buffer + BX], ' ' r
      INC CX
      MOV AL, [SI]
      INC SI
      CMP AL, 0Dh
      JBE process_text

finish:
```

```
    ; Print reversed text
    LEA DX, msg2
    MOV AH, 09h
    INT 21h

    LEA DX, rev_buffer
    MOV AH, 09h
    INT 21h

    ; Terminate program
    MOV AH, 4Ch
    INT 21h
Main ENDP
END main
```