INDEX

Part - A

Part - B

Part - A

1. Write an assembly language program to convert a lowercase letter to an uppercase letter or vice versa.

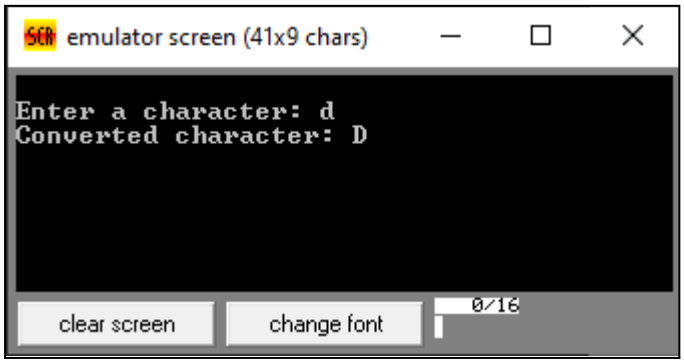
Program:

```
.MODEL SMALL
.STACK

.DATA
    MESSAGE DB 0DH, 0AH, 'Enter a character: $'
    RESULT DB 0DH, 0AH, 'Converted character: $'
    CHAR DB ?

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

    ; Display message asking for input
    MOV DX, OFFSET MESSAGE
    MOV AH, 09H
    INT 21H

    ; Read character from the user
    MOV AH, 01H
    INT 21H
    MOV CHAR, AL

    ; Check if the character is lowercase
    CMP CHAR, 'a'
    JB NOT_LOWER
    CMP CHAR, 'z'
    JA NOT_LOWER

    ; Convert lowercase to uppercase
    SUB CHAR, 32

    ; Jump to display result
    JMP DISPLAY_RESULT

NOT_LOWER:
    ; Check if the character is uppercase
    CMP CHAR, 'A'
    JB DISPLAY_RESULT
    CMP CHAR, 'Z'
    JA DISPLAY_RESULT

    ; Convert uppercase to lowercase
```

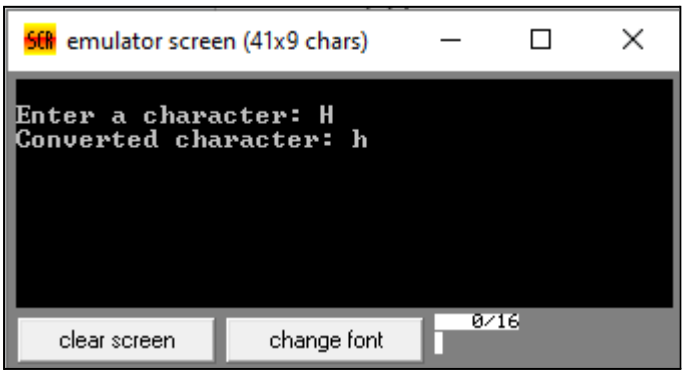
```
    ADD CHAR, 32

DISPLAY_RESULT:
    ; Display the converted character
    MOV DX, OFFSET RESULT
    MOV AH, 09H
    INT 21H

    MOV DL, CHAR
    MOV AH, 02H
    INT 21H

    ; Exit the program
    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

Output:

(lowercase to uppercase)



(uppercase to lowercase)



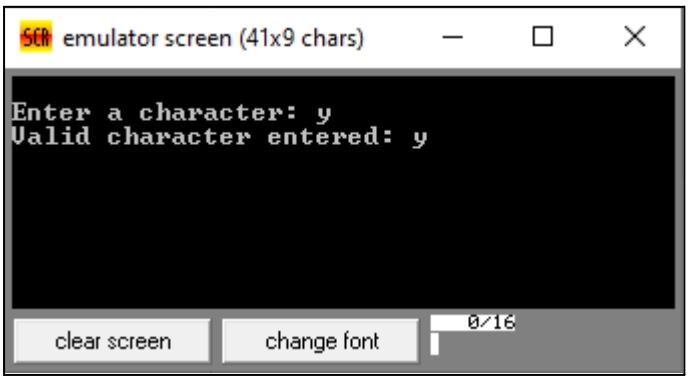
2. Write an assembly language program to read a character. If it is "y" or "Y", display it; otherwise terminate the program.

Program:

```
.MODEL SMALL
.STACK

.DATA
    MESSAGE DB 0DH, 0AH, 'Enter a character: $'
    VALID_MESSAGE DB 0DH, 0AH, 'Valid character entered: $'
    INVALID_MESSAGE DB 0DH, 0AH, 'Invalid character entered. Program terminated. $'
    CHAR DB ?

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

    ; Display message asking for input
    MOV DX, OFFSET MESSAGE
    MOV AH, 09H
    INT 21H

    ; Read character from the user
    MOV AH, 01H
    INT 21H
    MOV CHAR, AL

    ; Check if the character is 'y' or 'Y'
    CMP CHAR, 'y'
    JE DISPLAY_VALID
    CMP CHAR, 'Y'
    JE DISPLAY_VALID

    ; Display invalid character message and terminate
    MOV DX, OFFSET INVALID_MESSAGE
    MOV AH, 09H
    INT 21H
    JMP EXIT_PROGRAM

DISPLAY_VALID:
    ; Display the valid character
    MOV DX, OFFSET VALID_MESSAGE
    MOV AH, 09H
    INT 21H

    MOV DL, CHAR
    MOV AH, 02H
    INT 21H

EXIT_PROGRAM:
```

```
    ; Exit the program
    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

Output:



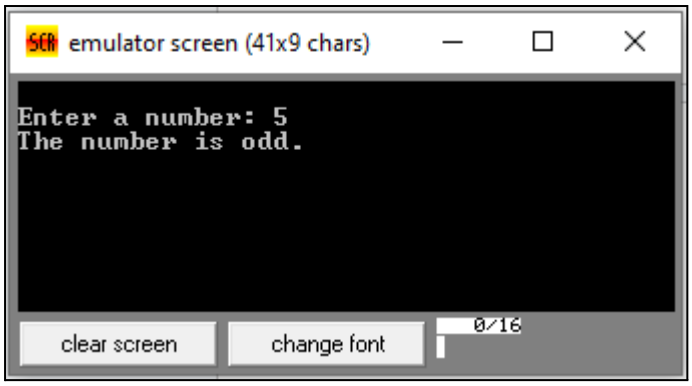
3. Write an assembly language program to determine whether a number is odd or even.

Program:

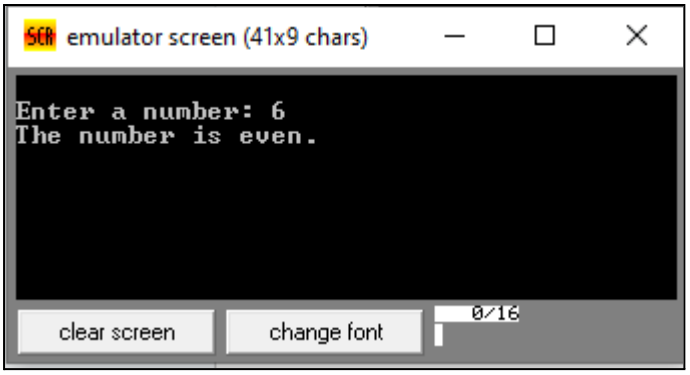
```
.MODEL SMALL
.STACK

.DATA
    MESSAGE DB 0DH, 0AH, 'Enter a number: $'
    ODD_MESSAGE DB 0DH, 0AH, 'The number is odd. $'
    EVEN_MESSAGE DB 0DH, 0AH, 'The number is even. $'
    NUMBER DB ?

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

    ; Display message asking for input
    MOV DX, OFFSET MESSAGE
    MOV AH, 09H
    INT 21H

    ; Read number from the user
    MOV AH, 01H
    INT 21H
    SUB AL, 30H  ; Convert ASCII digit to binary

    ; Check if the number is odd or even
    TEST AL, 01H  ; Perform bitwise AND with 00000001
    JNZ ODD  ; Jump if the result is not zero (odd)

    ; Number is even
    MOV DX, OFFSET EVEN_MESSAGE
    MOV AH, 09H
    INT 21H
    JMP EXIT_PROGRAM

ODD:
    ; Number is odd
    MOV DX, OFFSET ODD_MESSAGE
    MOV AH, 09H
    INT 21H

EXIT_PROGRAM:
    ; Exit the program
    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

Output:

(odd number detection)



(even number detection)



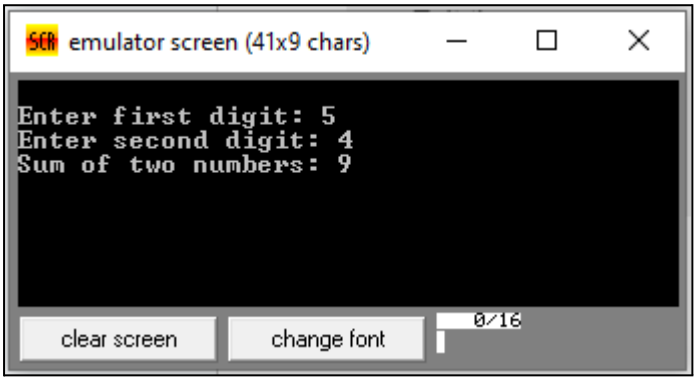
4. Write an assembly language program to add two decimal numbers.

Program:

```
.model small
.stack
.data
    val1 db ?
    val2 db ?
    sum db ?
    msg1 db 0dh, 0ah, "Enter first digit: $"
    msg2 db 0dh, 0ah, "Enter second digit: $"
    msg3 db 0dh, 0ah, "Sum of two numbers: $"

.code
main proc
    mov ax, @data
    mov ds, ax

    ; Read first digit
    mov dx, offset msg1
    mov ah, 9
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h  ; Convert ASCII digit to binary
    mov val1, al

    ; Read second digit
    mov dx, offset msg2
    mov ah, 9
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h  ; Convert ASCII digit to binary
    mov val2, al

    ; Add the digits
    mov al, val1
    add al, val2
    add al, 30h  ; Convert sum to ASCII

    ; Store the sum
    mov sum, al

    ; Display the sum
    mov dx, offset msg3
    mov ah, 9
    int 21h
```

```
    mov dl, sum
    mov ah, 2
    int 21h

    ; Exit the program
    mov ah, 4Ch
    int 21h
main endp
end main
```

Output:



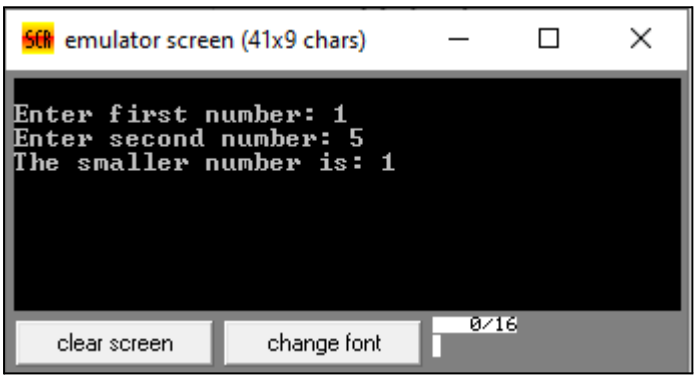
5. Write an assembly language program to input two numbers, compare them and display the smaller one.

Program:

```
.model small
.stack
.data
    num1 db ?
    num2 db ?
    msg1 db 0dh, 0ah, "Enter first number: $"
    msg2 db 0dh, 0ah, "Enter second number: $"
    msg3 db 0dh, 0ah, "The smaller number is: $"

.code
main proc
    mov ax, @data
    mov ds, ax

    ; Read first number
    mov dx, offset msg1
    mov ah, 9
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h  ; Convert ASCII digit to binary
    mov num1, al

    ; Read second number
    mov dx, offset msg2
    mov ah, 9
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h  ; Convert ASCII digit to binary
    mov num2, al

    ; Compare the numbers
    mov al, num1
    cmp al, num2
    jge num2_smaller

num1_smaller:
    ; Display num1 as the smaller number
    mov dx, offset msg3
    mov ah, 9
```

```
    int 21h

    mov dl, num1
    add dl, 30h  ; Convert to ASCII
    mov ah, 2
    int 21h
    jmp exit_program

num2_smaller:
    ; Display num2 as the smaller number
    mov dx, offset msg3
    mov ah, 9
    int 21h

    mov dl, num2
    add dl, 30h  ; Convert to ASCII
    mov ah, 2
    int 21h

exit_program:
    ; Exit the program
    mov ah, 4Ch
    int 21h

main endp
end main
```

Output:



Part - B

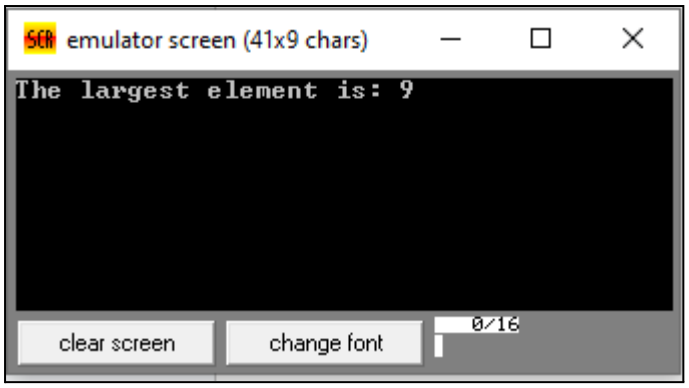
6. Write an assembly language program to find the largest element of an array.

Program:

```
.model small
.stack
.data
    array db 7, 5, 3, 9, 0, 1, 2
    size equ 7
    maxVal db 0
    msg db "The largest element is: $"

.code
main proc
    mov ax, @data
    mov ds, ax

    ; Initialize maxVal with the first element of the array
    mov al, array
    mov maxVal, al

    ; Loop through the array to find the largest element
    mov cx, size
    mov si, offset array + 1 ; Start from the second element

loop_start:
    mov al, byte ptr [si]
    cmp al, maxVal
    jle not_largest

    ; Update maxVal if a larger element is found
    mov maxVal, al

not_largest:
    inc si
    loop loop_start

    ; Display the largest element
    mov dx, offset msg
    mov ah, 09h
    int 21h

    mov dl, maxVal
    add dl, 30h  ; Convert to ASCII
    mov ah, 02h
    int 21h
```

```
    ; Exit the program
    mov ah, 4Ch
    int 21h

main endp
end main
```

Output:

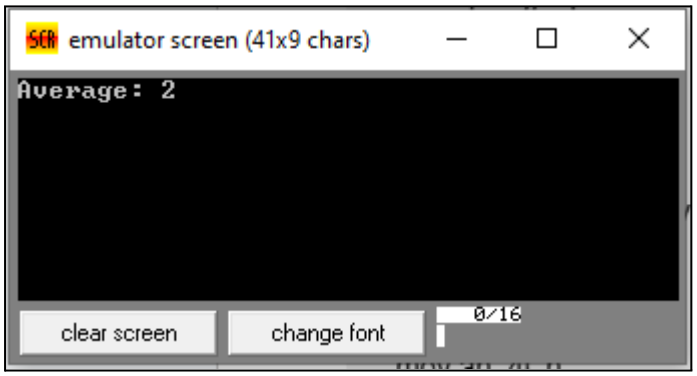


7. Write an assembly language program to calculate the average of numbers.

Program:

```
.model small
.stack
.data
    count dw 4        ; Number of elements in the array
    numbers db 2, 3, 1, 2   ; Example array with five elements
    sum dw 0          ; Variable to store the sum of numbers
    average db 0      ; Variable to store the average
    msg db "Average: $"
.code
main proc
    mov ax, @data
    mov ds, ax

    ; Calculate the sum of numbers
    mov cx, count
    mov si, offset numbers
    xor bx, bx

sum_loop:
    mov al, byte ptr [si]
    add bx, ax
    inc si
    loop sum_loop

    ; Calculate the average
    mov ax, bx
    cwd             ; Sign extend the 16-bit value in AX to DX:AX
    idiv count

    mov sum, bx        ; Store the sum
    mov average, al    ; Store the average

    ; Display the average
    mov dx, offset msg
    mov ah, 09h
    int 21h

    mov dl, average
    add dl, 30h        ; Convert the average to ASCII
    mov ah, 02h
    int 21h

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

Output:

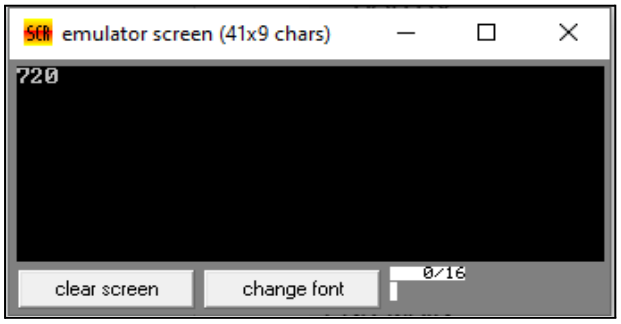

emulator screen (41x9 chars)
Average: 2
clear screen
change font
0/16

8. Write an assembly language program to calculate the factorial of an integer number.

Program:

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE

MAIN PROC

    MOV CX, 6
    MOV AX, 1

    TOP:
    MUL CX
    LOOP TOP

    MOV BX, 10
    MOV CX, 0

    CONVERT:
    XOR DX, DX
    DIV BX
    ADD DL, 30H
    PUSH DX
    INC CX
    OR AX, AX
    JNZ CONVERT

    DISPLAY:
    POP DX
    MOV AH, 02H
    INT 21H
    LOOP DISPLAY

    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

Output:



9. Write an assembly language program to sort numbers in ascending order.

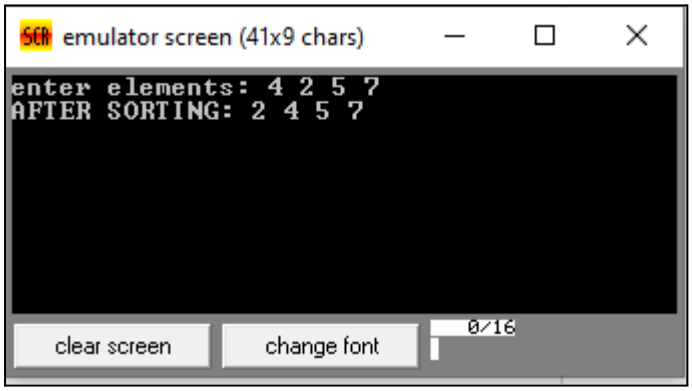
Program:

```
.MODEL SMALL
.STACK 100H
.DATA
MSG1 DB 'enter elements: $'
MSG2 DB 'AFTER SORTING: $'
ARR DB 100 dup (0)
.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX
MOV AH, 9
lea DX, MSG1    ; DISPLAY MSG1
INT 21H

XOR CX, CX
MOV AH, 1
INT 21H    ; first input
XOR SI, SI

WHILE_:
   CMP AL, 0dH    ; compare input with CR

   JE END_WHILE
   MOV ARR[SI], AL   ; move input into array
   INC SI    ;SI+1
   INC CX
   MOV AH, 2
   MOV DL, ' '    ; display space
   INT 21h
   MOV AH, 1
   INT 21H
JMP WHILE_

END_WHILE:
   MOV AH, 2
   MOV DL, 0DH
   INT 21H
   MOV DL, 0AH
   INT 21H
JCXZ EXIT
   LEA SI, ARR
   MOV BX, CX
CALL BUBBLE_SORT

MOV AH, 9
LEA DX, MSG2
INT 21H
```

```
XOR SI,SI
TOP:
   MOV AH, 2
   MOV DL, ARR[SI]
   INT 21H
   MOV DL, ' '
   INT 21H
   INC SI
   LOOP TOP
EXIT:
   MOV AH, 4CH
   INT 21H
   MAIN ENDP

BUBBLE_SORT PROC
   ; this procedure will sort the array in ascending order
   ; input : SI=offset address of the array
   ;       : BX=array size
   ; output: Sorted Array

   PUSH AX    ; push AX onto the STACK
   PUSH BX    ; push BX onto the STACK
   PUSH CX    ; push CX onto the STACK
   PUSH DX    ; push DX onto the STACK
   PUSH DI    ; push DI onto the STACK

   MOV AX, SI
   MOV CX, BX
   DEC CX

@OUTER_LOOP:
   MOV BX, CX
   MOV SI, AX
   MOV DI, AX
   INC DI

   @INNER_LOOP:
    MOV DL, [SI]
    CMP DL, [DI]
      JNG @SKIP_EXCHANGE

      XCHG DL, [DI]
      MOV [SI], DL
      @SKIP_EXCHANGE:
      INC SI
      INC DI
      DEC BX
   JNZ @INNER_LOOP
 LOOP @OUTER_LOOP

 POP DI
 POP DX
 POP CX
```

```
  POP BX
  POP AX

  RET

  BUBBLE_SORT ENDP

END MAIN
```

Output:



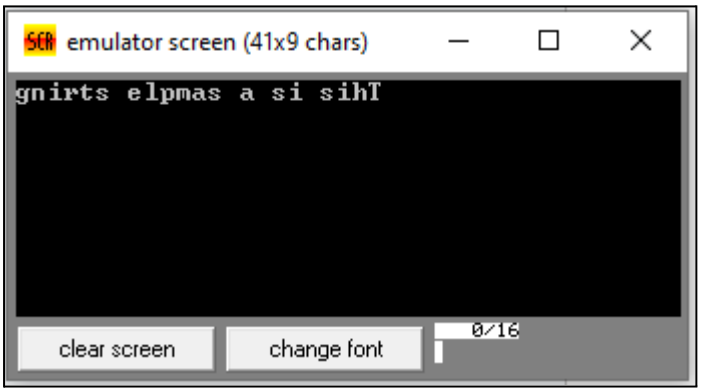
10. Write an assembly language program to accept a string from keyboard and display the string in reverse order.

Program:

```
.MODEL SMALL
.STACK 100H
.DATA

; The string to be printed
STRING DB 'This is a sample string', '$'

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

; call reverse function
CALL REVERSE

; load address of the string
LEA DX,STRING

; output the string
; loaded in dx
MOV AH, 09H
INT 21H

; interrupt to exit
MOV AH, 4CH
INT 21H

MAIN ENDP
REVERSE PROC
        ; load the offset of
        ; the string
        MOV SI, OFFSET STRING

        ; count of characters of the;
        ;string
        MOV CX, 0H

        LOOP1:
        ; compare if this is;
        ;the last character
        MOV AX, [SI]
        CMP AL, '$'
        JE LABEL1

        ; else push it in the;
        ;stack
        PUSH [SI]
```

```
        ; increment the pointer;
        ;and count
        INC SI
        INC CX

        JMP LOOP1

        LABEL1:
        ; again load the starting;
        ;address of the string
        MOV SI, OFFSET STRING

                LOOP2:
                ;if count not equal to zero
                CMP CX,0
                JE EXIT

                ; pop the top of stack
                POP DX

                ; make dh, 0
                XOR DH, DH

                ; put the character of the;
                ;reversed string
                MOV [SI], DX

                ; increment si and;
                ;decrement count
                INC SI
                DEC CX

                JMP LOOP2


        EXIT:
        ; add $ to the end of string
        MOV [SI],'$ '
        RET

REVERSE ENDP
END MAIN
```

Output:


emulator screen (41x9 chars)
gnirts elpmas a si sihT
clear screen
change font
0/16