AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Computer Science and Engineering
Program: BSc in Computer Science and Engineering

Course Code: CSE 2214

Assignment No: 04

Date of Submission: 17 August 2025

Submitted by,
Name: Abrar Ahmad Chy
Student ID: 202301004063
Lab Section: B1

Question 01: Write a program that lets the user type some text, consisting of words separated by blanks, ending with a carriage return, and displays the text in the same word order as entered, but with the letters in each word reversed.

Solution:

```
.MODEL SMALL
.STACK 100H

.DATA
  PROMPT1 DB 'Enter the string : $'
  PROMPT2 DB 0DH,0AH,'The string with words reversed : $'
  BUFFER  DB 100 DUP('$')   ; input buffer
  NEWLINE DB 0DH,0AH,'$'

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

    ; Show first prompt
    LEA DX, PROMPT1
    MOV AH, 9
    INT 21H

    ; Read input string
    XOR CX, CX
    LEA SI, BUFFER

READ_LOOP:
    MOV AH, 1          ; read char
    INT 21H
    CMP AL, 0DH        ; Enter pressed?
    JE END_INPUT
    MOV [SI], AL       ; store char
    INC SI
    INC CX
    JMP READ_LOOP

END_INPUT:
```

```
    MOV BYTE PTR [SI], 0   ; null-terminate string

    ; Show second prompt
    LEA DX, PROMPT2
    MOV AH, 9
    INT 21H

    ; Process words
    LEA SI, BUFFER

NEXT_WORD:
    ; Skip spaces
    MOV AL, [SI]
    CMP AL, 0
    JE DONE
    CMP AL, ' '
    JNE WORD_START

    MOV DL, ' '
    MOV AH, 2
    INT 21H
    INC SI
    JMP NEXT_WORD

WORD_START:
    MOV DI, SI
FIND_END:
    MOV AL, [DI]
    CMP AL, 0
    JE REVERSE
    CMP AL, ' '
    JE REVERSE
    INC DI
    JMP FIND_END

REVERSE:
    DEC DI
PRINT_BACK:
    CMP DI, SI
    JB WORD_DONE
```

```
    MOV DL, [DI]
    MOV AH, 2
    INT 21H
    DEC DI
    JMP PRINT_BACK

WORD_DONE:
    MOV AL, [SI]
    CMP AL, 0
    JE DONE
    ; move SI to next word
    FIND_SPACE:
        MOV AL, [SI]
        CMP AL, 0
        JE NEXT_WORD
        CMP AL, ' '
        JE NEXT_WORD
        INC SI
        JMP FIND_SPACE

DONE:
    ; Exit
    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

<u>Question 02:</u> Write a program that lets the user type in an algebraic expression, ending with a carriage return, that contains round (parentheses), square, and curly brackets. As the expression is being typed in, the program evaluates each character. If at any point the expression is

incorrectly bracketed (too many right brackets or a mismatch between left and right brackets), the program tells the user to start over. After the carriage return is typed, if the expression is correct, the program displays "expression is correct." If not, the program displays "too many left brackets". In both cases, the program asks the user if he or she wants to continue. If the user types 'Y', the program runs again. Your program does not need to store the input string, only check it for correctness.

Solution:

```
.MODEL SMALL
.STACK 100H

.DATA
  PROMPT          DB  0DH,0AH,'Enter an Algebraic Expression : $'
  CORRECT         DB  0DH,0AH,'Expression is Correct.$'
  LEFT_BRACKETS   DB  0DH,0AH,'Too many Left Brackets.$'
  RIGHT_BRACKETS  DB  0DH,0AH,'Too many Right Brackets. Begin Again!$'
  MISMATCH        DB  0DH,0AH,'Bracket Mismatch. Begin Again!$'
  CONTINUE        DB  0DH,0AH,'Type Y if you want to Continue : $'

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

@START:
```

```
    ; Show prompt
    LEA DX, PROMPT
    MOV AH, 9
    INT 21H

    XOR CX, CX          ; CX = stack depth (count of brackets)

@INPUT:
    MOV AH, 1
    INT 21H             ; read char into AL
    CMP AL, 0DH
    JE @END_INPUT       ; Enter pressed?

    ; --- Handle left brackets ---
    CMP AL, '('
    JE @PUSH
    CMP AL, '{'
    JE @PUSH
    CMP AL, '['
    JE @PUSH

    ; --- Handle right round bracket ---
    CMP AL, ')'
    JE @ROUND

    ; --- Handle right curly bracket ---
    CMP AL, '}'
    JE @CURLY

    ; --- Handle right square bracket ---
    CMP AL, ']'
    JE @SQUARE

    JMP @INPUT          ; ignore all other characters

@PUSH:                  ; push left bracket
    PUSH AX
    INC CX
    JMP @INPUT

@ROUND:                 ; found ")"
    CMP CX, 0
```

```
    JLE @RIGHT_BRACKETS   ; no matching "("
    POP DX
    DEC CX
    CMP DL, '('
    JNE @MISMATCH
    JMP @INPUT

@CURLY:                 ; found "}"
    CMP CX, 0
    JLE @RIGHT_BRACKETS
    POP DX
    DEC CX
    CMP DL, '{'
    JNE @MISMATCH
    JMP @INPUT

@SQUARE:                ; found "]"
    CMP CX, 0
    JLE @RIGHT_BRACKETS
    POP DX
    DEC CX
    CMP DL, '['
    JNE @MISMATCH
    JMP @INPUT

@END_INPUT:             ; Enter pressed
    CMP CX, 0
    JNE @LEFT_BRACKETS

    ; Expression is correct
    LEA DX, CORRECT
    MOV AH, 9
    INT 21H

    JMP @ASK_CONT

@MISMATCH:
    LEA DX, MISMATCH
    MOV AH, 9
    INT 21H
    JMP @ASK_CONT
```

```
@LEFT_BRACKETS:
    LEA DX, LEFT_BRACKETS
    MOV AH, 9
    INT 21H
    JMP @ASK_CONT

@RIGHT_BRACKETS:
    LEA DX, RIGHT_BRACKETS
    MOV AH, 9
    INT 21H
    JMP @ASK_CONT

@ASK_CONT:              ; ask user if they want to continue
    LEA DX, CONTINUE
    MOV AH, 9
    INT 21H

    MOV AH, 1
    INT 21H
    CMP AL, 'Y'
    JE @START

@EXIT:
    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```