**Lab Assignment # 02**

**COAL LAB – Spring 2022**

**Submission Deadline: Sunday 24th April 2022 @ 11:59 PM**

**Instructions Total Marks: 50**

| a. Programs must be MASM615 compatible.  b. Code must be commented properly.  c. You have to write your answers and screenshots of the output in this **.docx file.** d. For Part2, You have to submit all your codes in a **single .zip file;**  e. Submitted Filename must have format like **Assignment#\_Sec#\_Roll#** e.g. A1\_A\_20\_i1234 f. **Plagiarism will result in ZERO Marks in All Assignments of Class and Lab.** |
| --- |

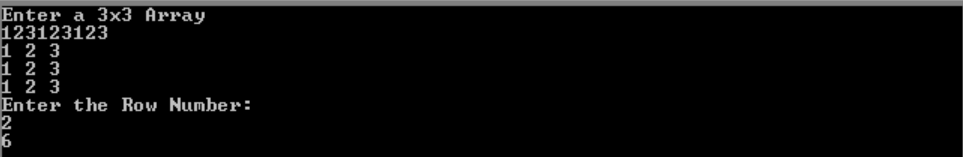
**Q1. Calculating the Sum of an Array Row**

Write a procedure named “calc\_row\_sum” that calculates the sum of a single row in a two-dimensional array of bytes or words. The procedure should have the following stack parameters: array offset, row size, array type, row index. It must return the sum in AX. Use explicit stack parameters. Write a program that tests your procedure with arrays of byte or word. Prompt the user for the row index, and display the sum of the selected row.

**( 15 Marks )**

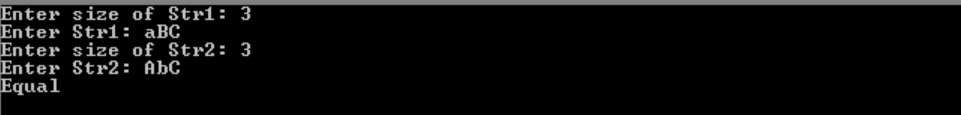
**Note:** Stack parameter means passing parameters through stack. Make sure you take inputs from the user.

| .model small .stack 100h .data arr db 9 dup(?) 1st db ? 2nd db ? 3rd db ?  variable db ? msg db "Enter the Row Number: $" smsg db "Enter a 3x3 Array $" emsg db "The sum of the row is: $" .code main proc  mov ax,@data mov ds,ax mov cx,9 mov si,offset arr  mov dl,offset smsg mov ah,09 int 21h call line call input ; calling input procedure call line ; calling line procedure to add line  mov si,offset arr mov cx,3 mov bx,0 l1: push cx mov cx,3 l2: mov dl,[si+bx] add dl,48 mov ah,02 int 21h inc si mov dl,32 mov ah,02 int 21h loop l2 call line ; calling line procedure  mov si,0 add bx,3 pop cx loop l1   mov dl,offset msg mov ah,9 int 21h call line mov ah,01 int 21h sub al,48 ; Adjusting the ascii mov variable,al call sum  exit: mov ah,4ch int 21h main endp  sum proc mov si,offset arr cmp al,1 ; FOR FIRST je ff  cmp al,2 ; FOR SECOND je seco  cmp al,3 ;FOR THIRD je th  mov al,'f' mov ah,02 int 21h jmp exr ff: mov cx,3 FIRST: mov al,[si] add 1st,al  inc si loop FIRST call line mov ah,02 add 1st,48 mov dl,1st int 21h call line jmp exr seco: add si,3 mov cx,3 second: mov al,[si] add 2nd,al inc si loop second call line mov ah,02 add 2nd,48 mov dl,2nd int 21h  mov dl,10 mov ah,02 int 21h  jmp exr th: add si,6 mov cx,3 third: mov al,[si] add 3rd,al inc si loop third call line mov ah,02 add 3rd,48 ; Stroing the answer in the accumulator register mov ah,0 mov al,3rd mov dl,al int 21h  call line exr: ret  sum endp ;Input Procedure  input proc  i: mov ah,01 int 21h sub al,48 mov [si],al  inc si loop i ret  input endp ;Line add procedure  line proc   mov dl,10 mov ah,02 int 21h mov dl,13 mov ah,02 int 21h ret line endp  end main |
| --- |



**Q2.** Write an assembly language procedure named “equalsIgnoreCase”, which receives two strings and their sizes, and returns true if the two strings contain the same characters irrespective of the case. For example, for strings {“aBc”} and {“Abc”} the function returns true, but for {“aBc”} and {“aB”}, or {“aBc”} and {“Xbz”}, the function returns false. Write a generic procedure that must handle all checks and conditions.

| .model small .stack 100h .data msg1 db "Enter size of Str1: $" smsg1 db "Enter Str1: $" msg2 db "Enter size of Str2: $" smsg2 db "Enter Str2: $" nmsg db "Not Equal $" smsg db "Equal $" linefeed db 13, 10, "$" len1 db ? len2 db ? str1 db len1 dup(?) str2 db len2 dup(?)  .code main proc  mov ax,@data mov ds,ax ;Length of String 1 mov dx, offset msg1 mov ah,9 int 21h mov ah,01 int 21h sub al,48 mov len1,al  ;String 1 Input mov ah, 09 mov dx, offset linefeed int 21h mov dx, offset smsg1 mov ah,9 int 21h  mov si, offset str1 mov cx,0 mov cl,len1  call input   ;Length of String 2 mov ah, 09 mov dx, offset linefeed int 21h  mov ax,0 mov dx, offset msg2 mov ah,9 int 21h mov ah,01 int 21h sub al,48 mov len2,al  ;String 2 Input mov ah, 09 mov dx, offset linefeed int 21h mov dx, offset smsg2 mov ah,9 int 21h mov si,0 mov si, offset str2 mov cx,0 mov cl,len2  call input   ;COMPARE LENGTH mov al,len1 cmp al,len2 JNE exit  ;CONVERTING1 mov si,0 mov si, offset str1 mov cx,0 mov cl,len1 l1: ;ascii of a mov al,[si] cmp al,'a' JL pass convert: sub [si],32 inc si pass: inc si loop l1  ;CONVERTING2 mov si,0 mov si, offset str2 mov cx,0 mov cl,len2 l2: ;ascii of a mov al,[si] cmp al,'a' JL pass1 convert1: sub [si],32 inc si pass1: inc si loop l2  ;FINALCOMPARE mov ax,0 mov si, offset str1 mov di, offset str2 mov cx,0 mov cl,len1 floop: mov al,[di] cmp [si],al JNE exit inc si inc di loop floop  s\_exit: mov ah, 09 mov dx, offset linefeed int 21h mov dx,offset smsg mov ah,9 int 21h mov ah,4ch int 21h   exit: mov ah, 09 mov dx, offset linefeed int 21h mov dx,offset nmsg mov ah,9 int 21h mov ah,4ch int 21h   main endp  input proc ;INPUT PROCEDURE  Input1: mov ah,01 int 21h mov [si],al mov al,0 loop Input1 ret  input endp  end main |
| --- |



**Q3.** Write a program to check whether the one string is substring of another string using procedures. **( 20 Marks )**

**.model small**

| .stack 100h .data msg1 db "Enter size of Main String: $" smsg1 db "Enter Main String: $" msg2 db "Enter size of Sub String: $" smsg2 db "Enter Sub String: $" nmsg db "Not Find $" smsg db "Found $" linefeed db 13, 10, "$" len1 db ? len2 db ? temp db ? str1 db len1 dup(?) str2 db len2 dup(?)  .code main proc  mov ax,@data mov ds,ax ;Length of String 1 call length mov len1,al ;String 1 Input mov ah, 09 mov dx, offset linefeed int 21h mov dx, offset smsg1 mov ah,9 int 21h mov si, offset str1 ; calling input procedure mov cx,0 mov cl,len1 call input    ;Length of String 2 call length mov len2,al ;String 2 Input mov ah, 09 mov dx, offset linefeed int 21h mov dx, offset smsg2 mov ah,9 int 21h mov si,0 mov si, offset str2 ; Calling input procedures mov cx,0 mov cl,len2 call input   ;Looping mov si,0 mov si, offset str1 mov di, offset str2 mov ch,0 mov cl,len1  mainl: mov al,[di] mov temp, offset di mov bl,[si] mov ch,0 mov cl,len2 CMP al,bl JE secl secl: mov dl,[si] cmp [di],dl JNE secl inc si inc di loop secl  main endp  input proc Input1: mov ah,01 int 21h mov [si],al mov al,0 loop Input1 ret  input endp  length proc mov ah, 09 mov dx, offset linefeed int 21h mov ax,0 mov dx, offset msg2 mov ah,9 int 21h mov ah,01 int 21h sub al,48 ret  length endp  end main |
| --- |

There are several ways to determine whether one string is a substring of another. The following way is probably the simplest. Suppose we declare.

SUBl DB 'ABC'

MAINST DB 'ABABCA'

and we want to see whether SUB1 and SUB2 arc substrings of MAINST. Let's begin with SUB1. We can compare corresponding characters in the strings.



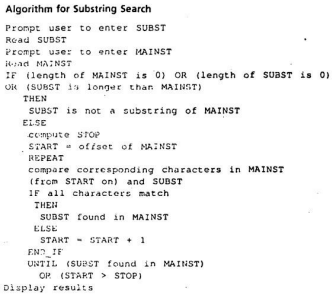
Because 'there is a mismatch at the third comparison, we backtrack and try to match SUBl with the part of MAINST from position MAINST+l on:



There is a mismatch immediately, so we begin again, and at position MAINST+2 

This 'time we are successful; SUBI is a substring of MAINST

Here is an Algorithm for Substring checking.



Test Case:

