

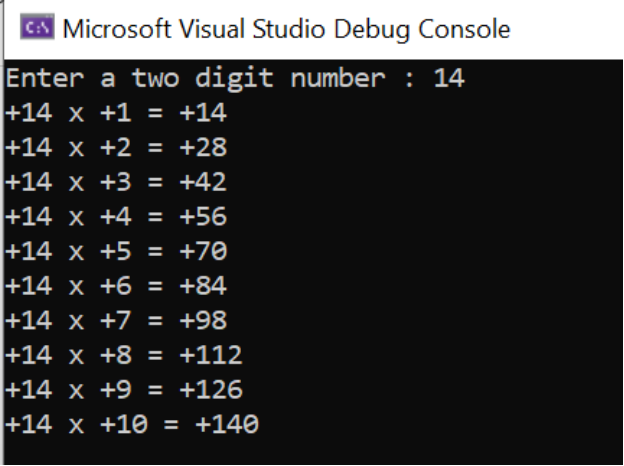
# ASSIGNMENT 3

Name: Kulsoom Khurshid

Reg#: Sp20-BCS-044

Course: Microprocessor and Assembly Language

**Question 1)** Write a program that asks user to input a two-digit integer and prints its multiple from 1 to 10. For example, if user gives 7 as input, the output must be:

<u>CODE</u>	<u>OUTPUT</u>
<pre>INCLUDE Irvine32.inc .data str1 byte "Enter a two digit number : ",0 val1 sdword ? str2 byte " x ",0 str3 byte " = ",0  .code main PROC     mov edx, offset str1     call WriteString     call ReadInt     mov val1, eax     mov ecx, 1     start:         cmp ecx, 10         jg terminate         mov eax, val1         call WriteInt         mov edx, offset str2         call WriteString         mov eax, ecx         call WriteInt         mov edx, offset str3         call WriteString         mov eax, val1         mul ecx         call WriteInt         call Crlf         inc ecx         jmp start     terminate:         exit main endp end main</pre>	 <p>Microsoft Visual Studio Debug Console</p> <pre>Enter a two digit number : 14 +14 x +1 = +14 +14 x +2 = +28 +14 x +3 = +42 +14 x +4 = +56 +14 x +5 = +70 +14 x +6 = +84 +14 x +7 = +98 +14 x +8 = +112 +14 x +9 = +126 +14 x +10 = +140</pre>

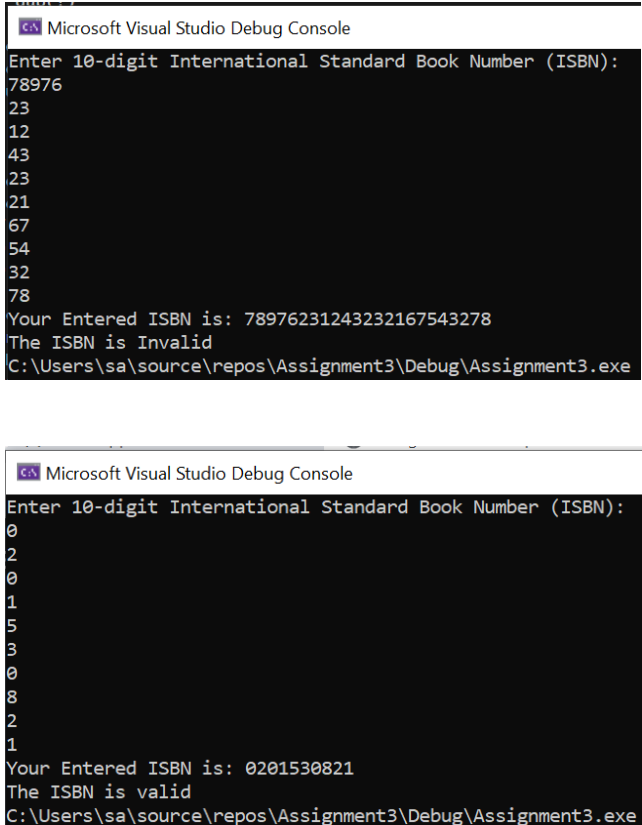
**Question 2)** Write an assembly language program that computes the Hamming distance between the two users provided strings. The Hamming distance is the number of bit positions where the two strings differ.

<u>CODE</u>	<u>OUTPUT</u>
<pre> include irvine32.inc .data str1 byte "Enter First String : ",0 str2 byte "Enter Second String : ",0 str3 byte "Hamming Distance is : ",0 val1 byte 20 DUP(0) val2 byte 20 DUP(0) arr1 byte 160 DUP(0) arr2 byte 160 DUP(0) val3 byte ? value dword ? counter dword ? counter1 dword ? counter2 dword ?  .code main proc     mov edx, offset str1     call WriteString     mov edx, OFFSET val1     mov ecx, SIZEOF val1- 1     call ReadString     mov ecx,0     mov esi,0     mov ebx,0     mov counter,0     start:         cmp ecx,20         jnl end_for         cmp val1[esi],0         jz end_for          push ecx         mov cl,0          start1:             cmp cl,8             jnl end_for1              movzx eax,val1[esi]             add cl,1             shr eax,cl             jc c1             jmp c2         c1:             mov arr1[ebx],1 </pre>	 

	<pre>         jmp c3 c2:      mov arr1[ebx],0         jmp c3 c3:      ;inc ecx         inc ebx         jmp start1  end_for1:         pop ecx         inc ecx         inc esi         jmp start end_for:         mov counter, ebx      mov edx, offset str2     call WriteString         mov edx, OFFSET val2     mov ecx, SIZEOF val2-1   call ReadString         mov ecx,0         mov esi,0         mov ebx,0         mov counter2,0 start5:  cmp ecx,20         jnl end_for5         cmp val2[esi],0         jz end_for5         push ecx         mov cl,0  start6:  cmp cl,8         jnl end_for6         movzx eax,val2[esi]         add cl,1         shr eax,cl         jc c4         jmp c5 c4:      mov arr2[ebx],1         jmp c6 c5:      mov arr2[ebx],0         jmp c6 c6:      ;inc ecx         inc ebx </pre>	
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

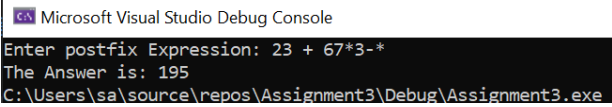
<pre>                 jmp start6              end_for6:                 pop ecx                 inc ecx                 inc esi                 jmp start5             end_for5:                 mov counter2, ebx                 mov eax,counter2                 cmp eax,counter                 jl c10                 mov eax,counter                 mov counter2,eax                 jmp c10             c10:                 mov counter1,0                 mov ecx,0                 mov eax,0             start4:                 cmp ecx,counter2                 jnl end_f                 movzx eax,arr1[ecx]                 movzx ebx,arr2[ecx]                 cmp eax,ebx                 je c7                 jmp c8             c7:                 add counter1,1             c8:                 inc ecx                 jmp start4             end_f:                 mov edx, offset str3                 call WriteString                 mov eax , counter2                 sub eax,counter1                 call WriteDec                 call crlf                 exit main endp end main </pre>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

**Question 3)** Write an assembly language program that validates a user provided 10-digit International Standard Book Number (ISBN). For an ISBN to be valid, the following weighted sum modulo 11 must be equal to 0.

<u>CODE</u>	<u>OUTPUT</u>
<pre> INCLUDE Irvine32.inc .data str1 byte "Enter 10-digit International Standard Book Number (ISBN) (hint: enter one digit and then press enter): ",0 val1 dword 10 dup(?) sum dword ? str2 byte "The ISBN is valid ",0 str3 byte "The ISBN is Invalid ",0 str4 byte "Your Entered ISBN is: ",0  .code main PROC     mov edx, offset str1     call WriteString     call Crlf     mov esi,0     mov ecx,0     start:         cmp ecx,10         jnb end_for         call ReadInt         mov val1[esi],eax         add esi,4         inc ecx         jmp start     end_for:         mov edx, offset str4         call WriteString         mov ecx,0         mov esi,0         mov ebx,1         mov sum,0     start1:         cmp ecx,10         jnb end_for1         mov eax,val1[esi]         call Writedec         mul ebx         add sum,eax         add esi,4         inc ecx         inc ebx         jmp start1     end_for1:         call Crlf         mov eax,sum         mov ebx,11         mov edx,0         div ebx </pre>	 <p>The output shows two separate runs of the program in the Microsoft Visual Studio Debug Console. In the first run, the user enters '78976' as the ISBN, and the program outputs 'The ISBN is Invalid'. In the second run, the user enters '0201530821' as the ISBN, and the program outputs 'The ISBN is valid'.</p>

<pre> mov eax,edx cmp eax,0 je l1 mov edx, offset str3 call WriteString jmp l2 l1: mov edx, offset str2 call WriteString jmp l2 l2: exit main endp end main </pre>	
--------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

**Question 4)** Write an assembly language program that evaluates a user provided postfix expression and displays the result on console.

<u>CODE</u>	<u>OUTPUT</u>
<pre> INCLUDE Irvine32.inc .data str1 byte "Enter postfix Expression: ",0 str2 byte "The Answer is: ",0 val1 byte 20 DUP(?) val2 dword ?  .code main PROC     mov edx, OFFSET str1     call writestring     mov edx, OFFSET val1     mov ecx, SIZEOF val1-1     call ReadString     mov ecx,0     mov esi,0     start:         cmp ecx,SIZEOF val1-1         jnle end_for         movzx eax,val1[esi] ; get character         cmp eax,'1'         je l1         cmp eax,'2'         je l2         cmp eax,'3'         je l3         cmp eax,'4'         je l4         cmp eax,'5'         je l5         cmp eax,'6'         je l6         cmp eax,'7' </pre>	 <p>Microsoft Visual Studio Debug Console</p> <p>Enter postfix Expression: 23 + 67*3-*</p> <p>The Answer is: 195</p> <p>C:\Users\sa\source\repos\Assignment3\Debug\Assignment3.exe</p>

	je l7 cmp eax,'8' je l8 cmp eax,'9' je l9 cmp eax,'+' je l10 cmp eax,'-' je l11 cmp eax,'*' je l12 cmp eax,'/' je l13 jmp l14	
l1:	mov eax,1 push eax jmp l14	
l2:	mov eax,2 push eax jmp l14	
l3:	mov eax,3 push eax jmp l14	
l4:	mov eax,4 push eax jmp l14	
l5:	mov eax,5 push eax jmp l14	
l6:	mov eax,6 push eax jmp l14	
l7:	mov eax,7 push eax jmp l14	
l8:	mov eax,8 push eax jmp l14	
l9:	mov eax,9 push eax jmp l14	
l10:	pop eax mov val2,eax pop eax add eax,val2 push eax jmp l14	

<pre>l11:    pop eax         mov val2,eax         pop eax         sub eax,val2         push eax         jmp l14  l12:    pop eax         mov val2,eax         pop eax         mov ebx,val2         mul ebx         push eax         jmp l14  l13:    pop eax         mov val2,eax         pop eax         mov ebx,val2         xor edx, edx         div ebx         push eax         jmp l14  l14:    inc esi         inc ecx         jmp start end_for: mov edx, OFFSET str2 call writestring pop eax call writedec exit  main endp end main</pre>	
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--