

**Comp 3350: Computer Organization & Assembly Language**  
**HW # 9: Theme: Advanced Procedures, Stack Parameters, Locals and BCD**  
*(All main questions carry equal weight. Credit awarded to only those answers for which work has been shown.)*

---

1. Write a procedure named *ArraySeries* that fills an array of ten (10) numbers with the Fibonacci series. The procedure receives two arguments: the first is the offset of an array, and the second is an integer that specifies the array length. The first argument is passed by value and the second is passed by reference. In the main program, you should set the parameters of the array and print the array values before and after call to the procedure.

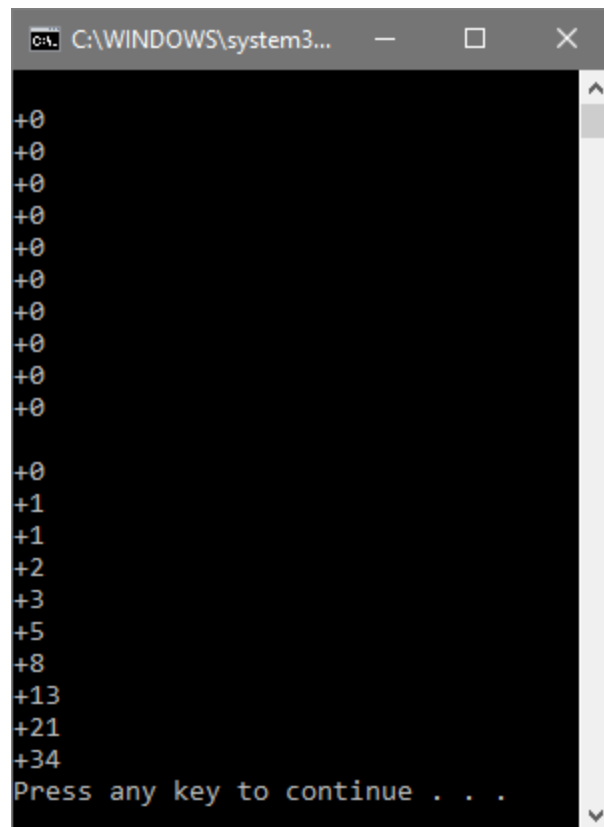
Please embed your code into your homework solution along with a screen shot of the run of the program.

```
TITLE
INCLUDE Irvine32.inc
.data
count = 10
fibArray WORD 0 , 9 DUP(?)
.code
main PROC
push OFFSET fibArray
push count
call DisplayArray

push OFFSET fibArray
push count
call ArraySeries

push OFFSET fibArray
push count
call DisplayArray
exit
main ENDP

ArraySeries PROC
push ebp
mov ebp, esp
mov esi, [ebp+12]
mov ecx, [ebp+8]
mov ebx, 0
L2:
mov eax, [esi + ebx]
add eax, [esi + ebx - 2]
add ebx, TYPE fibArray
mov [esi + ebx], eax
.IF ecx > 9
add eax, 1
mov [esi + ebx], eax
.ENDIF
loop L2
pop ebp
ret
ArraySeries ENDP
```



```
C:\WINDOWS\system32\cmd.exe
+0
+0
+0
+0
+0
+0
+0
+0
+0
+0
+0
+1
+1
+2
+3
+5
+8
+13
+21
+34
Press any key to continue . . .
```

```

DisplayArray PROC
push ebp
mov ebp, esp
mov esi, [ebp+12]
mov ecx, [ebp+8]
mov ebx, 0
L1:
movsx eax, WORD PTR [esi + ebx]
add ebx, TYPE fibArray
call Crlf
call WriteInt
loop L1
call Crlf
pop ebp
ret 8
DisplayArray ENDP

END main

```

2. Draft a program that adds two BCD numbers (10-digits each). The first BCD number is stored in an array named *myAuburnID*, and the second in an array named *myAurbunIdRev*. The first number is your actual Auburn ID (with a prefix single zero digit and the remaining digits as the 9-digits of your *Auburn ID*); the second is the value of *MyAuburnId* written backwards. Your program should do the following:

- 1) Use shifts/rotates using *myAuburnID* to fill the array *myAuburnIdRev*
- 2) Display contents of the memory locations in question
- 3) Add *myAuburnID* and *myAurbunIdRev* using BCD arithmetic
- 4) Store the sum in a variable named *Result*, and
- 5) Display contents of memory post execution.

Please embed your code into your homework solution along with a screen shot post execution.

```

TITLE
INCLUDE Irvine32.inc
.data
myAuburnID BYTE 0h, 9h, 0h, 3h, 9h, 0h, 7h, 9h, 7h, 7h
myAuburnIDRev BYTE 10 DUP(?)
Result BYTE 10 DUP(0)
MSG1 BYTE "Result", 0
MSG2 BYTE "MyAuburnID", 0
.code
main PROC
mov esi, 0
mov ecx, 10
L3:
mov al, myAuburnID[esi]
mov edx, 9
sub edx, esi
mov myAuburnIDRev[edx], al
inc esi
loop L3

mov edx, OFFSET MSG2
call WriteString

```

```

mov edx, 0
call CrLf
push OFFSET myAuburnID
push 10
call DisplayArray
call CrLf

mov edx, OFFSET MSG1
call WriteString
mov edx, 0

call CrLf
mov ecx, 10
mov edx, 9
mov esi, OFFSET myAuburnID
mov edi, OFFSET myAuburnIDRev
mov ebx, OFFSET Result

L1:
    mov al, [esi + edx]
    mov ah, [edi + edx]
    add al, ah
    daa
    mov byte ptr [ebx + edx], al
    dec edx
loop L1

push OFFSET Result
push 10
CALL DisplayArray
exit
main ENDP

DisplayArray PROC
push ebp
mov ebp, esp
mov esi, [ebp+12]
mov ecx, [ebp+8]
mov ebx, 0

L2:
movzx eax, BYTE PTR [esi + ebx]
add ebx, TYPE BYTE
CALL WriteHex
call CrLf
Loop L2
pop ebp
ret 8
DisplayArray ENDP

END main

```

```

C:\WINDOWS\system...
MyAuburnID
00000000
00000009
00000000
00000003
00000009
00000000
00000007
00000009
00000007
00000007

Result
00000007
00000016
00000009
00000010
00000009
00000009
00000010
00000009
00000016
00000007

Press any key to continue . . .

```

3. Consider an isosceles triangle A with base 12 and height 20. Consider another triangle B formed using vertices which are the center of the sides of triangle A. Consider another triangle C whose vertices are similarly formed from B. Repeat this process ad infinitum. Express the sum of the areas of all such triangles using a series and its closed form sum. Compute the areas (a) by using

only the first two terms of the series and (b) by using the closed form of the series sum. Write a program to find the sums and use shifts to compute. What is the difference in the two computed sums?

Please embed your code into your homework solution along with a screen shot post execution.

```
TITLE
INCLUDE Irvine32.inc
.data
height DWORD 20
base    DWORD 12
msg1    BYTE "The sum of the first two of the series is: ", 0
msg2    BYTE "The sum of the series is: ", 0
msg3    BYTE "The difference in the two sums is: ", 0
.code
main PROC
call SumOfA
call SumOfB
call Diff
main ENDP

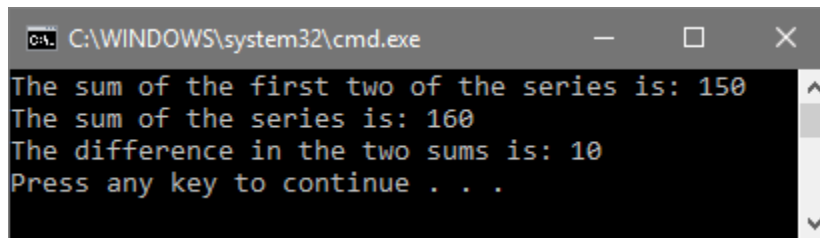
SumOfA PROC
mov eax, height
mov ebx, base
mul ebx
shr eax, 1 ; A = bh / 2
mov ebx, 5
mul ebx
shr eax, 2 ; A + A / 4 = 5 * A / 4
mov edx, OFFSET msg1
call WriteString
call WriteDec
call Crlf
SumOfA ENDP

SumOfB PROC
push eax
mov ebx, 5
xor edx, edx
div ebx
shl eax, 2 ; A = ((5 * A / 4) / 5) * 4

shl eax, 2
mov ebx, 3
xor edx, edx
div ebx ; A / (1 - 1 / 4) = 4 * A / 3
mov edx, OFFSET msg2
mov ebx, eax
call WriteString
call WriteDec
call Crlf
pop eax
SumOfB ENDP

Diff PROC
sub ebx, eax
```

```
mov eax, ebx
mov edx, OFFSET msg3
call WriteString
call WriteDec
call Crlf
Diff ENDP
    Exit
END main
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\WINDOWS\system32\cmd.exe". The window contains the following text output from an assembly program:

```
The sum of the first two of the series is: 150
The sum of the series is: 160
The difference in the two sums is: 10
Press any key to continue . . .
```

The text is displayed in a monospaced font on a black background. A vertical scrollbar is visible on the right side of the window.