Comp 3350: Computer Organization & Assembly Language
HW # 7: Theme: Conditionals, Booleans, Loops

(All main questions carry equal weight. Credit awarded to only those answers for which work has been shown.)

1. Draft a program that scans an array to determine the first positive EVEN number in the array. If a positive value is found, the program should print "positive even number found" and the value. If no positive EVEN value is found in the array, the program should print "no positive even number found." Submit list file and show the runs for the following data items:
    a. all negative even values
    b. all positive odd values
    c. mixed negative and positive values which are odd and even (two different examples with odd and even numbers at different indices)

a.

```
; Author: Zejian Zhong
; UserID: zzz0065

TITLE POSITIVE EVEN
INCLUDE Irvine32.inc

.data
myArray1 SWORD -2,-4,-6,-8
sentinel SWORD 0
message1 BYTE "positive even number found",0
message2 BYTE "no postive even number found",0

.code
main PROC
mov ecx, LENGTHOF myArray1
mov ebx, 0

L1:
mov ax, [myArray1 + ebx]
mov dl, 2
idiv dl
cmp ah, 0
jz L2
add ebx, TYPE myArray1
LOOP L1

MSG2:
mov edx, OFFSET message2
CALL WriteString
CALL Crlf  ≤1ms elapsed
jmp exitThis

L2:
mov ax, [myArray1 + ebx]
test ax, 8000h
```

C:\Users\nic05\Desktop\7\7.1\Debug\Project.exe

```
no postive even number found
```

b.

```
; Author: Zejian Zhong
; UserID: zzz0065

TITLE POSITIVE EVEN
INCLUDE Irvine32.inc

.data
myArray1 SWORD 1,3,5,7
sentinel SWORD 0
message1 BYTE "positive even number found",0
message2 BYTE "no postive even number found",0

.code
main PROC
mov ecx, LENGTHOF myArray1
mov ebx, 0

L1:
mov ax, [myArray1 + ebx]
mov dl, 2
idiv dl
cmp ah, 0
jz L2
add ebx, TYPE myArray1
LOOP L1

MSG2:
mov edx, OFFSET message2
CALL WriteString
CALL Crlf
jmp exitThis

L2:
mov ax, [myArray1 + ebx]
test ax, 8000h
```

C:\Users\nic05\Desktop\7\7.1\Debug\Project.exe

```
no postive even number found
```

c.Mix1

```
; Author: Zejian Zhong
; UserID: zzz0065

TITLE POSITIVE EVEN
INCLUDE Irvine32.inc

.data
myArray1 SWORD -1,2,-3,5,7,-11
sentinel SWORD 0
message1 BYTE "positive even number found",0
message2 BYTE "no postive even number found",0

.code
main PROC
mov ecx, LENGTHOF myArray1
mov ebx, 0

L1:
mov ax, [myArray1 + ebx]
mov dl, 2
idiv dl
cmp ah, 0
jz L2
add ebx, TYPE myArray1
LOOP L1

MSG2:
mov edx, OFFSET message2
CALL WriteString
CALL Crlf
jmp exitThis

L2:
mov ax, [myArray1 + ebx]
test ax, 8000h
```

C:\Users\nic05\Desktop\7\7.1\Debug\Project.exe

```
positive even number found
```

c.Mix2

```
; Author: Zejian Zhong
; UserID: zzz0065

TITLE POSITIVE EVEN
INCLUDE Irvine32.inc

.data
myArray1 SWORD -1,-3,5,7,2,-11
sentinel SWORD 0
message1 BYTE "positive even number found",0
message2 BYTE "no postive even number found",0

.code
main PROC
mov ecx, LENGTHOF myArray1
mov ebx, 0

L1:
mov ax, [myArray1 + ebx]
mov dl, 2
idiv dl
cmp ah, 0
jz L2
add ebx, TYPE myArray1
LOOP L1

MSG2:
mov edx, OFFSET message2
CALL WriteString
CALL Crlf
jmp exitThis

L2:
mov ax, [myArray1 + ebx]
test ax, 8000h
```

```
C:\Users\nic05\Desktop\7\7.1\Debug\Project.exe
positive even number found
```

2. Write a program which encodes any string using the XOR instruction. Test it using your <first name last name> in the data segment to produce cipher text and then decode using the program to get plain text. Use the last three digits of your student id as the key. Print plane text from the data segment, print the cipher text, and then print the plain text upon execution. What are the strengths and weaknesses of this encryption method? Can you think of another way doing such encryption? What are the strengths and weaknesses of your method?

```
C:\WINDOWS\system32\cmd.exe
Enter the plain text: zejian zhong

Cipher text:          ¼╗╗╗│╝≥¿│╝╝╝╢

Decrypted:            zejian zhong

Press any key to continue . . .
```

Strengths: Hard to find out original text at one glance
Weaknesses: The length of cipher text is the same with original text, the original text can be guessed in long run. Using only one key will also increase the risk.

Other method: One time pad encryption
Strengths: Every key is one-time used, more secure.
Weakness: Cannot be decode if losing the key pad.

3. Implement the following two pseudo-codes in assembly language (assume signed numbers). Declare Apple and Pear as word sized variables. Test the program for input data sets listed below and print values assigned to Apple and Pear. Submit list file and show output for all input data sets.

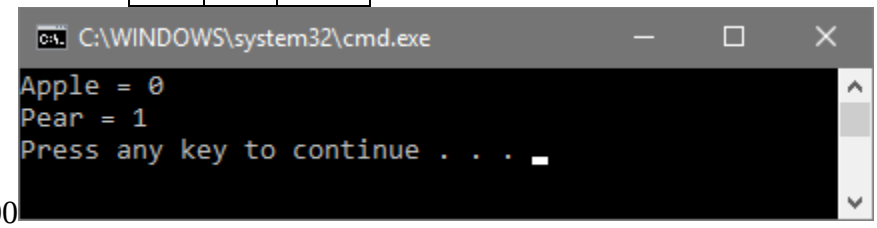
A.
```
if ( (cx = bx) AND (cx >=  val1) )
    Apple = 1;
Else
    Apple = 0;
```

B.
```
if  ( (bx = cx)  OR  (cx >= val1) )
    Pear = 1;
Else
    Pear = 0;
```

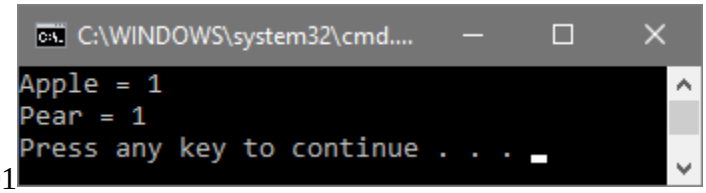
Input test data

CX	BX	Val1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

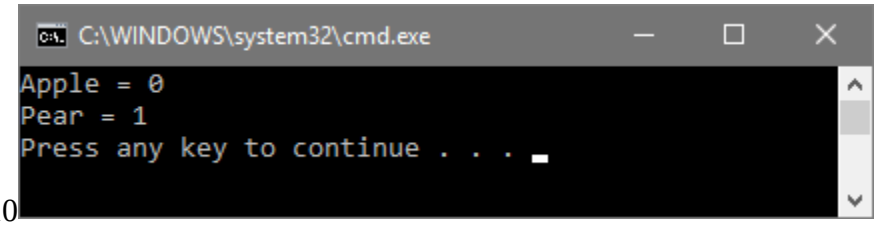
000



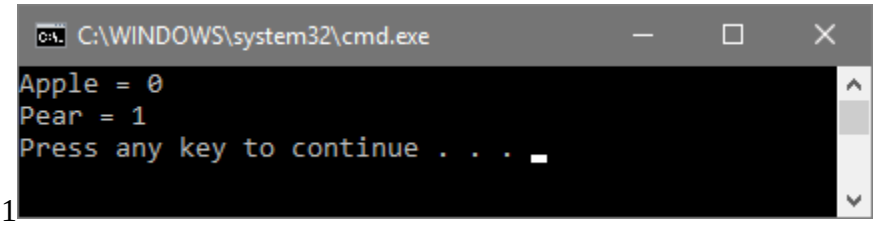
001



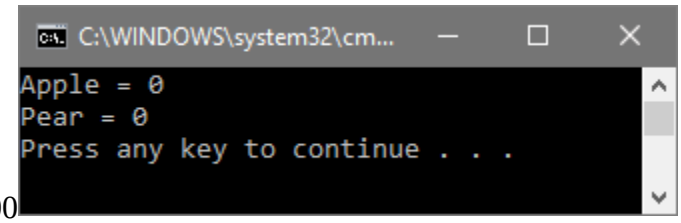
010

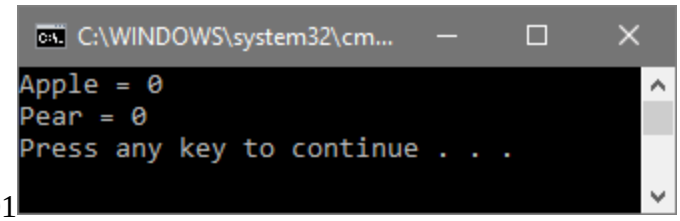


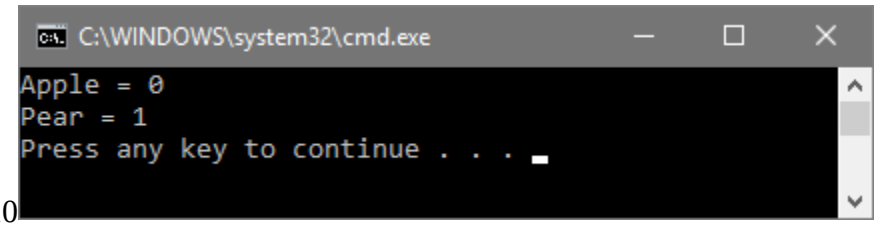
011

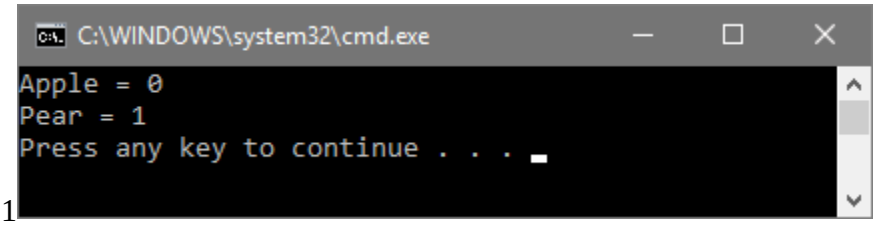


100



101 

110 

111 

4. Draw the stack (pencil-paper or word→pdf) at different points of the main and subroutine to show your understanding of the call and return functions.

```
        Main PROC
4040040     call FloatAdd
4040046     mov eax, ebx
        …
        …

        FloatAdd  PROC
4041020     Push ecx
4041024     Push ebx
4041028     mov eax, edx
        …
        …
404A030     Pop ebx
404A032     Pop ecx
404A034     ret
        FloatAdd ENDP
```

Before call to FloatAdd

OFFSET		
00001000		ESP
00000FFC	00000000	
00000FF8	00000000	
00000FF4	00000000	
00000FF0	00000000	

EIP = 04040040

After call to FloatAdd

OFFSET		
00001000		ESP
00000FFC	04040046	
00000FF8	00000000	
00000FF4	00000000	
00000FF0	00000000	

EIP = 04041020

After Push ecx

OFFSET		
00001000		
00000FFC	04040046	
00000FF8	(ecx)	ESP
00000FF4	00000000	
00000FF0	00000000	

After Push ebx

OFFSET		
00001000		
00000FFC	04040046	
00000FF8	(ecx)	
00000FF4	(ebx)	ESP
00000FF0	00000000	

After Pop ebx

OFFSET		
00001000		
00000FFC	04040046	
00000FF8	(ecx)	ESP
00000FF4	00000000	
00000FF0	00000000	

After Pop ecx

OFFSET		
00001000		ESP
00000FFC	04040046	
00000FF8	00000000	

00000FF4	00000000
00000FF0	00000000

After Return

OFFSET		
00001000		ESP
00000FFC	00000000	
00000FF8	00000000	
00000FF4	00000000	
00000FF0	00000000	

EIP = 04040046