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 alphanumeric array to determine the first alphabet in the array. If found, the program should print “alphabet found” its value and index. If no alphabet is found, the program should print “no alphabet found.” **Submit the asm/list file and screenshots that show the output of your code for the following example arrays:**

a. Array has only alphabets; Boundary Case

b. Array has only numerals; Boundary Case

c. Several arrays with a mix of alphabets and numerals positioned at different indices

e.g. MyArray BYTE 67, -3, 74, G, W, 92 ;General Case

Value = 'G'

Index = 3

```
;hw7-1.asm

INCLUDE Irvine32.inc

.data

MyArray1 BYTE  'a', 'B', 'c', 'D', 'e', 'F'
;MyArray1 BYTE  44, 3, 8, 22, 12, 34
;MyArray1 BYTE  67, -3, 74, 'G', 'W',92
;MyArray1 BYTE  5, 'H', 74, 'G', 'W',92
;MyArray1 BYTE  4, 2, 11, 'r', 'W',92
;MyArray1 BYTE  10, 3, 'J', -22, 'W',92
message BYTE "alphabet found! ", 0
message1 BYTE " Index: ", 0
message2 BYTE "Value: ", 0
message3 BYTE "No alphabet found!", 0

.code
main PROC

mov esi, OFFSET MyArray1
mov ecx, LENGTHOF MyArray1
mov esi, 0

L1: cmp esi, ecx
jl L2
jmp NotFound

L2:
cmp MyArray1[esi], 'A' ; if(MyArray[esi] >= 'A')
```

```
jge L3
jb  L6



L3:
cmp MyArray1[esi], 'Z'
jle Found
jg L4


L4:
cmp MyArray1[esi], 'a'
jge L5
jb L6


L5:
cmp MyArray1[esi], 'z'
jle Found
jg L6

L6:
       inc esi
       jmp L1



Found:
mov al, MyArray1[esi]
mov edx, OFFSET message
call WriteString
mov edx, OFFSET message2
call WriteString
mov al, MyArray1[esi]
call WriteChar
mov edx, OFFSET message1
call WriteString
mov eax, esi
call WriteDec
jmp quit

NotFound:
mov edx, OFFSET message3
call WriteString

quit:
call Crlf
exit

invoke ExitProcess,0
main endp
end main
```

hw7-1.lst:

```
;hw7-1.asm

                            INCLUDE Irvine32.inc
                           C ; Include file for Irvine32.lib             (Irvine32.inc)
                           C
                           C ;OPTION CASEMAP:NONE             ; optional: make
identifiers case-sensitive
                           C
                           C INCLUDE SmallWin.inc             ; MS-Windows prototypes,
structures, and constants
                           C .NOLIST
                           C .LIST
                           C
                           C INCLUDE VirtualKeys.inc
                           C ; VirtualKeys.inc
                           C .NOLIST
                           C .LIST
                           C
                           C
                           C .NOLIST
                           C .LIST
                           C

 00000000                   .data

 00000000 61 42 63 44 65    MyArray1 BYTE  'a', 'B', 'c', 'D', 'e', 'F'
          46
                            ;MyArray1 BYTE  44, 3, 8, 22, 12, 34
                            ;MyArray1 BYTE  67, -3, 74, 'G', 'W',92
                            ;MyArray1 BYTE  5, 'H', 74, 'G', 'W',92
                            ;MyArray1 BYTE  4, 2, 11, 'r', 'W',92
                            ;MyArray1 BYTE  10, 3, 'J', -22, 'W',92
 00000006 61 6C 70 68 61    message BYTE "alphabet found! ", 0
          62 65 74 20 66
          6F 75 6E 64 21
          20 00
 00000017 20 49 6E 64 65    message1 BYTE " Index: ", 0
          78 3A 20 00
 00000020 56 61 6C 75 65    message2 BYTE "Value: ", 0
          3A 20 00
 00000028 4E 6F 20 61 6C    message3 BYTE "No alphabet found!", 0
          70 68 61 62 65
          74 20 66 6F 75
          6E 64 21 00

 00000000                   .code
 00000000                   main PROC

 00000000  BE 00000000 R    mov esi, OFFSET MyArray1
 00000005  B9 00000006             mov ecx, LENGTHOF MyArray1
```

```
0000000A  BE 00000000            mov esi, 0

0000000F  3B F1            L1: cmp esi, ecx
00000011  7C 02            jl L2
00000013  EB 67            jmp NotFound

00000015                   L2:
00000015  80 BE 00000000 R cmp MyArray1[esi], 'A' ; if(MyArray[esi] >= 'A')
         41
0000001C  7D 02            jge L3
0000001E  72 21            jb  L6


00000020                   L3:
00000020  80 BE 00000000 R cmp MyArray1[esi], 'Z'
         5A
00000027  7E 1B            jle Found
00000029  7F 00            jg L4


0000002B                   L4:
0000002B  80 BE 00000000 R cmp MyArray1[esi], 'a'
         61
00000032  7D 02            jge L5
00000034  72 0B            jb L6


00000036                   L5:
00000036  80 BE 00000000 R cmp MyArray1[esi], 'z'
         7A
0000003D  7E 05            jle Found
0000003F  7F 00            jg L6

00000041                   L6:
00000041  46                     inc esi
00000042  EB CB                  jmp L1



00000044                   Found:
00000044  8A 86 00000000 R mov al, MyArray1[esi]
0000004A  BA 00000006 R    mov edx, OFFSET message
0000004F  E8 00000000 E    call WriteString
00000054  BA 00000020 R    mov edx, OFFSET message2
00000059  E8 00000000 E    call WriteString
0000005E  8A 86 00000000 R mov al, MyArray1[esi]
00000064  E8 00000000 E    call WriteChar
00000069  BA 00000017 R    mov edx, OFFSET message1
0000006E  E8 00000000 E    call WriteString
00000073  8B C6            mov eax, esi
00000075  E8 00000000 E    call WriteDec
0000007A  EB 0A            jmp quit

0000007C                   NotFound:
0000007C  BA 00000028 R    mov edx, OFFSET message3
00000081  E8 00000000 E    call WriteString

00000086                   quit:
```

```
 00000086  E8 00000000 E    call Crlf
                            exit
 0000008B  6A 00        *       push   +000000000h
 0000008D  E8 00000000 E   *          call   ExitProcess

                            invoke ExitProcess,0
 00000092  6A 00        *       push   +000000000h
 00000094  E8 00000000 E   *          call   ExitProcess
 00000099                   main endp
                            end main
_Microsoft (R) Macro Assembler Version 14.28.29337.0       03/11/21 15:25:57
hw7-1.asm                                             Symbols 2 - 1
```

Snapshots:

a. Array has only alphabets; Boundary Case

```
;MyArray1 BYTE  'a', 'B', 'c', 'D', 'e', 'F'
```

Microsoft Visual Studio Debug Console

```
alphabet found! Value: a Index: 0

C:\Irvine\Examples\Project32\Debug\Project.exe (process 23844) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when
Press any key to close this window . . .
```

b. Array has only numerals; Boundary Case

```
;MyArray1 BYTE  44, 3, 8, 22, 12, 34
```

Microsoft Visual Studio Debug Console

```
No alphabet found!

C:\Irvine\Examples\Project32\Debug\Project.exe (process 17340) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

c. Several arrays with a mix of alphabets and numerals positioned at different indices

```
;MyArray1 BYTE  67, -3, 74, 'G', 'W',92
```

Microsoft Visual Studio Debug Console

```
alphabet found! Value: C Index: 0

C:\Irvine\Examples\Project32\Debug\Project.exe (process 21468) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

```
;MyArray1 BYTE  5, 'H', 74, 'G', 'W',92
```

Microsoft Visual Studio Debug Console

```
alphabet found! Value: H Index: 1

C:\Irvine\Examples\Project32\Debug\Project.exe (process 14220) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

```
;MyArray1 BYTE  4, 2, 11, 'r', 'W',92
```

Microsoft Visual Studio Debug Console

```
alphabet found! Value: r Index: 3

C:\Irvine\Examples\Project32\Debug\Project.exe (process 13252) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging st
Press any key to close this window . . .
```

```
;MyArray1 BYTE  10, 3, 'J', -22, 'W',92
```

Microsoft Visual Studio Debug Console

```
alphabet found! Value: J Index: 2

C:\Irvine\Examples\Project32\Debug\Project.exe (process 22748) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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 two 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. **Submit the asm/list file and screenshots that shows the output of your code.**

```
; Encryption Program               (hw7-2.asm)

; This program demonstrates simple symmetric
; encryption using the XOR instruction.

INCLUDE Irvine32.inc
KEY = 09      ; any value between 1-255
BUFMAX = 128         ; maximum buffer size

.data
sPrompt  BYTE  "Enter plain text: ",0
sEncrypt BYTE  "Cipher text:          ",0
sDecrypt BYTE  "Decrypted:            ",0
buffer   BYTE   BUFMAX+1 DUP(0)
bufSize  DWORD  ?

.code
main PROC
       call   InputTheString              ; input the plain text
       call   TranslateBuffer      ; encrypt the buffer
       mov    edx,OFFSET sEncrypt  ; display encrypted message
       call   DisplayMessage
       call   TranslateBuffer      ; decrypt the buffer
       mov    edx,OFFSET sDecrypt  ; display decrypted message
       call   DisplayMessage

       exit
main ENDP

;-----------------------------------------------------
InputTheString PROC
;
; Prompts user for a plaintext string. Saves the string
; and its length.
; Receives: nothing
; Returns: nothing
;-----------------------------------------------------
       pushad
       mov    edx,OFFSET sPrompt   ; display a prompt
       call   WriteString
       mov    ecx,BUFMAX           ; maximum character count
       mov    edx,OFFSET buffer   ; point to the buffer
       call   ReadString           ; input the string
       mov    bufSize,eax          ; save the length
       call   Crlf
       popad
       ret
```

```
InputTheString ENDP

;-----------------------------------------------------
DisplayMessage PROC
;
; Displays the encrypted or decrypted message.
; Receives: EDX points to the message
; Returns:  nothing
;-----------------------------------------------------
       pushad
       call   WriteString
       mov    edx,OFFSET buffer    ; display the buffer
       call   WriteString
       call   Crlf
       call   Crlf
       popad
       ret
DisplayMessage ENDP

;-----------------------------------------------------
TranslateBuffer PROC
;
; Translates the string by exclusive-ORing each
; byte with the encryption key byte.
; Receives: nothing
; Returns: nothing
;-----------------------------------------------------
       pushad
       mov    ecx,bufSize          ; loop counter
       mov    esi,0                ; index 0 in buffer
L1:
       xor    buffer[esi],KEY      ; translate a byte
       inc    esi                        ; point to next byte
       loop   L1

       popad
       ret
TranslateBuffer ENDP
END main
```

hw7-2.lst:

```
Microsoft (R) Macro Assembler Version 14.28.29337.0        03/11/21 15:45:42
hw7-2.asm                                             Page 1 - 1


                            ; Encryption Program               (hw7-2.asm)

                            ; This program demonstrates simple symmetric
                            ; encryption using the XOR instruction.

                            INCLUDE Irvine32.inc
                           C ; Include file for Irvine32.lib            (Irvine32.inc)
                           C
                           C ;OPTION CASEMAP:NONE             ; optional: make
identifiers case-sensitive
                           C
                           C INCLUDE SmallWin.inc             ; MS-Windows prototypes,
structures, and constants
                           C .NOLIST
                           C .LIST
                           C
                           C INCLUDE VirtualKeys.inc
                           C ; VirtualKeys.inc
                           C .NOLIST
                           C .LIST
                           C
                           C
                           C .NOLIST
                           C .LIST
                           C
 = 00000009                 KEY = 09      ; any value between 1-255
 = 00000080                 BUFMAX = 128         ; maximum buffer size

 00000000                   .data
 00000000 45 6E 74 65 72    sPrompt  BYTE  "Enter plain text: ",0
          20 70 6C 61 69
          6E 20 74 65 78
          74 3A 20 00
 00000013 43 69 70 68 65    sEncrypt BYTE  "Cipher text:          ",0
          72 20 74 65 78
          74 3A 20 20 20
          20 20 20 20 20
          20 20 00
 0000002A 44 65 63 72 79    sDecrypt BYTE  "Decrypted:            ",0
          70 74 65 64 3A
          20 20 20 20 20
          20 20 20 20 20
          20 20 00
 00000041  00000081 [              buffer   BYTE   BUFMAX+1 DUP(0)
           00
          ]
 000000C2 00000000          bufSize  DWORD  ?

 00000000                   .code
 00000000                   main PROC
```

```
 00000000  E8 00000025                    call   InputTheString             ; input the
plain text
 00000005  E8 00000062                    call   TranslateBuffer      ; encrypt the buffer
 0000000A  BA 00000013 R          mov    edx,OFFSET sEncrypt  ; display encrypted message
 0000000F  E8 0000003C                    call   DisplayMessage
 00000014  E8 00000053                    call   TranslateBuffer      ; decrypt the buffer
 00000019  BA 0000002A R          mov    edx,OFFSET sDecrypt  ; display decrypted message
 0000001E  E8 0000002D                    call   DisplayMessage

                                  exit
 00000023  6A 00        *       push   +000000000h
 00000025  E8 00000000 E   *          call   ExitProcess
 0000002A                  main ENDP

                           ;-----------------------------------------------------
 0000002A                  InputTheString PROC
                           ;
                           ; Prompts user for a plaintext string. Saves the string
                           ; and its length.
                           ; Receives: nothing
                           ; Returns: nothing
                           ;-----------------------------------------------------
 0000002A  60                     pushad
 0000002B  BA 00000000 R          mov    edx,OFFSET sPrompt   ; display a prompt
 00000030  E8 00000000 E          call   WriteString
 00000035  B9 00000080                    mov    ecx,BUFMAX           ; maximum character
count
 0000003A  BA 00000041 R          mov    edx,OFFSET buffer   ; point to the buffer
 0000003F  E8 00000000 E          call   ReadString           ; input the string
 00000044  A3 000000C2 R          mov    bufSize,eax          ; save the length
 00000049  E8 00000000 E          call   Crlf
 0000004E  61                     popad
 0000004F  C3                     ret
 00000050                  InputTheString ENDP

                           ;-----------------------------------------------------
 00000050                  DisplayMessage PROC
                           ;
                           ; Displays the encrypted or decrypted message.
                           ; Receives: EDX points to the message
                           ; Returns:  nothing
                           ;-----------------------------------------------------
 00000050  60                     pushad
 00000051  E8 00000000 E          call   WriteString
 00000056  BA 00000041 R          mov    edx,OFFSET buffer    ; display the buffer
 0000005B  E8 00000000 E          call   WriteString
 00000060  E8 00000000 E          call   Crlf
 00000065  E8 00000000 E          call   Crlf
 0000006A  61                     popad
 0000006B  C3                     ret
 0000006C                  DisplayMessage ENDP

                           ;-----------------------------------------------------
 0000006C                  TranslateBuffer PROC
                           ;
                           ; Translates the string by exclusive-ORing each
                           ; byte with the encryption key byte.
                           ; Receives: nothing
```

```
                          ; Returns: nothing
                          ;-----------------------------------------------------
 0000006C  60                     pushad
 0000006D  8B 0D 000000C2 R       mov    ecx,bufSize          ; loop counter
 00000073  BE 00000000                   mov    esi,0                ; index 0 in buffer
 00000078                 L1:
 00000078  80 B6 00000041 R       xor    buffer[esi],KEY      ; translate a byte
          09
 0000007F  46                     inc    esi                         ; point to next byte
 00000080  E2 F6                  loop   L1

 00000082  61                     popad
 00000083  C3                     ret
 00000084                 TranslateBuffer ENDP
                          END main
_Microsoft (R) Macro Assembler Version 14.28.29337.0       03/11/21 15:45:42
hw7-2.asm                                             Symbols 2 - 1
```

Screenshot of output after inputting <first name last name> as plain text:

Microsoft Visual Studio Debug Console

```
Enter plain text: Daniel Benjamin

Cipher text:         Mhg`le)Klgchd`g

Decrypted:           Daniel Benjamin


C:\Irvine\Examples\Project32\Debug\Project.exe (process 2968) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when de
Press any key to close this window . . .
```

What are the strengths and weaknesses of this encryption method (**25% of points, Typewritten answer required**)?

Strengths of this encryption method:

For this simple encryption method, the same single-character encryption key is used for both encryption and decryption. Mostly referred to as symmetric encryption or secret key cryptography, it is primarily used to ensure the confidentiality of a single user's important data.

Weaknesses of this encryption method:

This encryption method, known as symmetric encryption, uses a single-character encryption key. When encrypting important data, a single-character encryption key should never be used since it can be too easily decoded. Encryption keys containing multiple characters should be used to encrypt and decrypt plain text, since the length of the encryption key directly correlates with an encryption algorithm's strength. Also, the same single-character encryption key is used for both encryption and decryption; meaning that the sender and receiver of the encryption key must be the same person to ensure privacy and confidentiality.

3. Write a program that gets its input from two sensors. If the values of the sensors differ by no more than +/- 4, print "Agree", otherwise, print "Disagree." You can assume that the values are integers. Additionally, if the values Agree and they are each more than 50, print "Take Action". **Submit asm/list file and show screenshots of robust testing for various inputs, including boundary conditions, in the closed interval (-70 … 70).**

```
;hw7-3.asm

INCLUDE Irvine32.inc

.data
str1 BYTE "Value of sensor 1: ", 0
str2 BYTE "Value of sensor 2: ", 0
message BYTE "Difference of Sensor Values: ", 0
message1 BYTE "  Agree ", 0
message2 BYTE " Disagree", 0
message3 BYTE " Take Action", 0
errmess BYTE " Values differ by more than +/- 4! ", 0
errmess1 BYTE "Sensor values of 1 and 2 are greater than equal to 50! ", 0
sens1Val DWORD ?
sens2Val DWORD ?
diff DWORD ?

.code
main PROC
mov edx, OFFSET str1
call WriteString
call ReadInt
mov sens1Val, eax

mov edx, OFFSET str2
call WriteString
call ReadInt
```

```
;Finding difference for the values of Sensors 1 and 2

mov edx, OFFSET message
call WriteString

mov sens2Val, eax
mov eax, sens1Val
sub eax, sens2Val
mov diff, eax
call WriteInt

cmp eax, +4
jle lowBound
jg error

lowBound:
cmp eax, -4
jge accept
jl error


accept:
mov edx, OFFSET message1
call WriteString
mov eax, 50
cmp sens1Val, eax
jge checkSens2Val
exit

checkSens2Val:
mov eax, 50
cmp sens2Val, eax
jge sensValslargerthanfifty

sensValslargerthanfifty:
call Crlf
mov edx, OFFSET errmess1
call WriteString
mov edx, OFFSET message3
call WriteString
exit


error:
mov edx, OFFSET errmess
call WriteString

mov edx, OFFSET message2
call WriteString
exit



invoke ExitProcess,0
main ENDP
end main
```

hw7-3.lst:

```
;hw7-3.asm

                            INCLUDE Irvine32.inc
                           C ; Include file for Irvine32.lib            (Irvine32.inc)
                           C
                           C ;OPTION CASEMAP:NONE             ; optional: make
identifiers case-sensitive
                           C
                           C INCLUDE SmallWin.inc             ; MS-Windows prototypes,
structures, and constants
                           C .NOLIST
                           C .LIST
                           C
                           C INCLUDE VirtualKeys.inc
                           C ; VirtualKeys.inc
                           C .NOLIST
                           C .LIST
                           C
                           C
                           C .NOLIST
                           C .LIST
                           C

 00000000                   .data
 00000000 56 61 6C 75 65    str1 BYTE "Value of sensor 1: ", 0
          20 6F 66 20 73
          65 6E 73 6F 72
          20 31 3A 20 00
 00000014 56 61 6C 75 65    str2 BYTE "Value of sensor 2: ", 0
          20 6F 66 20 73
          65 6E 73 6F 72
          20 32 3A 20 00
 00000028 44 69 66 66 65    message BYTE "Difference of Sensor Values: ", 0
          72 65 6E 63 65
          20 6F 66 20 53
          65 6E 73 6F 72
          20 56 61 6C 75
          65 73 3A 20 00
 00000046 20 20 41 67 72    message1 BYTE "  Agree ", 0
          65 65 20 00
 0000004F 20 44 69 73 61    message2 BYTE " Disagree", 0
          67 72 65 65 00
 00000059 20 54 61 6B 65    message3 BYTE " Take Action", 0
          20 41 63 74 69
          6F 6E 00
 00000066 20 56 61 6C 75    errmess BYTE " Values differ by more than +/- 4! ", 0
          65 73 20 64 69
          66 66 65 72 20
          62 79 20 6D 6F
          72 65 20 74 68
          61 6E 20 2B 2F
          2D 20 34 21 20
          00
 0000008A 53 65 6E 73 6F    errmess1 BYTE "Sensor values of 1 and 2 are greater than equal
to 50! ", 0
          72 20 76 61 6C
```

```
         75 65 73 20 6F
         66 20 31 20 61
         6E 64 20 32 20
         61 72 65 20 67
         72 65 61 74 65
         72 20 74 68 61
         6E 20 65 71 75
         61 6C 20 74 6F
         20 35 30 21 20
         00
000000C2 00000000          sens1Val DWORD ?
000000C6 00000000          sens2Val DWORD ?
000000CA 00000000          diff DWORD ?

00000000                   .code
00000000                   main PROC
00000000  BA 00000000 R    mov edx, OFFSET str1
00000005  E8 00000000 E    call WriteString
0000000A  E8 00000000 E    call ReadInt
0000000F  A3 000000C2 R    mov sens1Val, eax

00000014  BA 00000014 R    mov edx, OFFSET str2
00000019  E8 00000000 E    call WriteString
0000001E  E8 00000000 E    call ReadInt


                           ;Finding difference for the values of Sensors 1 and 2

00000023  BA 00000028 R    mov edx, OFFSET message
00000028  E8 00000000 E    call WriteString

0000002D  A3 000000C6 R    mov sens2Val, eax
00000032  A1 000000C2 R    mov eax, sens1Val
00000037  2B 05 000000C6 R sub eax, sens2Val
0000003D  A3 000000CA R    mov diff, eax
00000042  E8 00000000 E    call WriteInt

00000047  83 F8 04         cmp eax, +4
0000004A  7E 02            jle lowBound
0000004C  7F 52            jg error

0000004E                   lowBound:
0000004E  83 F8 FC         cmp eax, -4
00000051  7D 02            jge accept
00000053  7C 4B            jl error




00000055                   accept:
00000055  BA 00000046 R    mov edx, OFFSET message1
0000005A  E8 00000000 E    call WriteString
0000005F  B8 00000032             mov eax, 50
00000064  39 05 000000C2 R cmp sens1Val, eax
0000006A  7D 07            jge checkSens2Val
                           exit
0000006C  6A 00        *       push   +000000000h
```

```
0000006E  E8 00000000 E   *          call   ExitProcess

00000073                  checkSens2Val:
00000073  B8 00000032            mov eax, 50
00000078  39 05 000000C6 R cmp sens2Val, eax
0000007E  7D 00           jge sensValslargerthanfifty

00000080                  sensValslargerthanfifty:
00000080  E8 00000000 E   call Crlf
00000085  BA 0000008A R   mov edx, OFFSET errmess1
0000008A  E8 00000000 E   call WriteString
0000008F  BA 00000059 R   mov edx, OFFSET message3
00000094  E8 00000000 E   call WriteString
                          exit
00000099  6A 00        *      push   +000000000h
0000009B  E8 00000000 E   *          call   ExitProcess





000000A0                  error:
000000A0  BA 00000066 R   mov edx, OFFSET errmess
000000A5  E8 00000000 E   call WriteString

000000AA  BA 0000004F R   mov edx, OFFSET message2
000000AF  E8 00000000 E   call WriteString
                          exit
000000B4  6A 00        *      push   +000000000h
000000B6  E8 00000000 E   *          call   ExitProcess



                          invoke ExitProcess,0
000000BB  6A 00        *      push   +000000000h
000000BD  E8 00000000 E   *          call   ExitProcess
000000C2                  main ENDP
                          end main
```

Screenshots:

Boundary conditions:

- Two sensor values that are the same:

```
Microsoft Visual Studio Debug Console
Value of sensor 1: 4
Value of sensor 2: 4
Difference of Sensor Values: +0  Agree
C:\Irvine\Examples\Project32\Debug\Project.exe (process 18984) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- Two sensor values that have a difference of -4:

```
Microsoft Visual Studio Debug Console
Value of sensor 1: 1
Value of sensor 2: 5
Difference of Sensor Values: -4  Agree
C:\Irvine\Examples\Project32\Debug\Project.exe (process 808) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debu
Press any key to close this window . . .
```

- Two sensor values that have a difference of +4:

```
Microsoft Visual Studio Debug Console
Value of sensor 1: 8
Value of sensor 2: 4
Difference of Sensor Values: +4  Agree
C:\Irvine\Examples\Project32\Debug\Project.exe (process 2212) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- Two sensor values that have a difference greater than +4:

```
Microsoft Visual Studio Debug Console
Value of sensor 1: 8
Value of sensor 2: 3
Difference of Sensor Values: +5 Values differ by more than +/- 4!  Disagree
C:\Irvine\Examples\Project32\Debug\Project.exe (process 1384) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- Two sensor values that have a difference greater than -4:

Microsoft Visual Studio Debug Console

```
Value of sensor 1: 1
Value of sensor 2: 6
Difference of Sensor Values: -5 Values differ by more than +/- 4!  Disagree
C:\Irvine\Examples\Project32\Debug\Project.exe (process 14948) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- Two sensor values that are both equal to +50 and have a difference less than +4:

Microsoft Visual Studio Debug Console

```
Value of sensor 1: 50
Value of sensor 2: 50
Difference of Sensor Values: +0  Agree
Sensor values of 1 and 2 are both greater than equal to 50!  Take Action
C:\Irvine\Examples\Project32\Debug\Project.exe (process 28564) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- Two sensor values that both exceed +50 and have a difference less than +4:

Microsoft Visual Studio Debug Console

```
Value of sensor 1: 51
Value of sensor 2: 51
Difference of Sensor Values: +0  Agree
Sensor values of 1 and 2 are greater than 50!  Take Action
C:\Irvine\Examples\Project32\Debug\Project.exe (process 19216) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- Two sensor values that both exceed -50 and have a difference less than +4:

Microsoft Visual Studio Debug Console

```
Value of sensor 1: -51
Value of sensor 2: -51
Difference of Sensor Values: +0  Agree
C:\Irvine\Examples\Project32\Debug\Project.exe (process 18960) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

4. Draw the stack (use, word/pdf) before every instruction that is marked red is executed to show your understanding of the call and return functions. Use N/A to represent unpredictable values.

```
Main Proc


        1:  4040018          mov ecx, 0000000Ch
        2:  404001C          mov ebx, 0000000Bh
        3:  4040020          call FADD
        4:  4040026          mov eax, ebx
                    …
                    …

          Main EndP

          FADD  PROC
       5: 4041040          Push ecx
       6: 4041044          Push ebx
       7:   4041048          mov eax, edx
                    …
                    …
       8:   404A060         Pop ebx
       9:  404A062         Pop ecx
       10:   404A064         ret
          FADD EndP
```

Stack on next page:

Instructions

0x4040018	1: mov ecx, 0000000Ch ; ECX = 0000000Ch
0x404001C	2: mov ebx, 0000000Bh ; EBX = 0000000Bh
0x4040020	3: call FADD - EIP points to to next instruction at 0x4041040 within FADD PROC - 0x4040026 is now stored as ESP

Whenever a **call** is made, the following process takes place inside the microprocessor:
- the address of the next instruction that exists in the caller program (after the program call instruction) is stored in the stack.
-the instruction queue is emptied for accommodating the instructions of the procedure.

contents of the instruction pointer (IP) is changed with the address of the first instruction of the procedure.

subsequent instructions of the procedure are stored in the instruction queue for execution.

0x4040026	4: mov eax, ebx ; EAX = 0000000Bh
0x4041040	5: push ecx - esp is decremented by 4 and the value of ecx (0000000Ch), is pushed onto the stack at 0x4041022
0x4041044	6: push ebx - esp is decremented by 4 and the value of ebx (0000000Bh), is pushed onto the stack at 0x404101E
0x4041048	7: mov eax, edx ; EAX = N/A (EDX is unknown based on the sample code since it has not been previously defined)
0x404A060	8: Pop ebx - esp is incremented by 4 and the value of ebx (0000000Bh), is popped off the stack at 0x404101E and esp now points to 0x4041022
0x404A062	9: Pop ecx - esp is incremented by 4 and the value of ecx (0000000Ch), is popped off the stack at 0x4041022 and esp now points to 0x4041026
0x404A064	10: ret - eip is now stored as 0x4040026

Stack after excecution of push instructions

0x4040026	N/A	
0x4041022	0000000Ch	
0x404101E	0000000Bh	<- ESP

Stack after excecution of pop instructions

0x4040026	N/A	<- ESP
0x4041022	-------	
0x404101E	-------	