**HW # 6:  Stacks and Procedures**

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

1. [Procedures] Write a main program which sets the registers BX to either 0, 1 or 2.  Write a  procedure *DisplayTiger which*will display the string "War Eagle" in Blue or Orange depending upon the whether the input is 0 or 1.  Single step through the program, displaying the values of the stack pointer so that you understand how the call and return are implemented.

List file for hw6-1.asm

Microsoft (R) Macro Assembler Version 14.28.29337.0 03/03/21 19:03:00

hw6-1.asm Page 1 - 1

;hw6-1.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

; declare variables here

00000000 45 6E 74 65 72 prompt BYTE "Enter an integer value to display War Eagle in either Blue or Orange: Blue(0), Orange(1)", 0

20 61 6E 20 69

6E 74 65 67 65

72 20 76 61 6C

75 65 20 74 6F

20 64 69 73 70

6C 61 79 20 57

61 72 20 45 61

67 6C 65 20 69

6E 20 65 69 74

68 65 72 20 42

6C 75 65 20 6F

72 20 4F 72 61

6E 67 65 3A 20

42 6C 75 65 28

30 29 2C 20 4F

72 61 6E 67 65

28 31 29 00

00000059 57 61 72 20 45 str1 BYTE "War Eagle", 0

61 67 6C 65 00

= 00000071 BlueTextOnGray = blue + (lightGray \* 16)

= 00000007 DefaultColor = lightGray + (black \* 16)

00000000 .code

00000000 main proc

00000000 BA 00000000 R mov edx, OFFSET prompt

00000005 E8 00000000 E call WriteString

0000000A E8 00000000 E call Crlf

0000000F E8 00000000 E call ReadInt

00000014 8B D8 mov ebx, eax

00000016 E8 00000010 call DisplayTiger

0000001B 66| B8 0007 mov ax, DefaultColor

0000001F E8 00000000 E call SetTextColor

invoke ExitProcess,0

00000024 6A 00 \* push +000000000h

00000026 E8 00000000 E \* call ExitProcess

0000002B main endp

;-----------------------------------------------------

0000002B DisplayTiger proc USES ebx

;

; Will display the string "War Eagle" in Blue or Orange depending upon the whether the input is 0 or 1.

; Receives: EAX = 0 | 1 | 2 , EBX = 0 | 1 | 2

; Returns: Returns the string "War Eagle" in either in Blue or Orange depending upon the whether the EBX register value is 0 or 1.

;-----------------------------------------------------

0000002B 53 \* push ebx

0000002C 83 FB 00 cmp ebx, 0

0000002F 74 1C JZ printBlue

00000031 75 00 JNZ printOrange

00000033 printOrange:

00000033 83 FB 02 cmp ebx, 2

00000036 74 13 JZ return

00000038 66| B8 0084 mov ax, red + (gray \* 16)

0000003C E8 00000000 E call SetTextColor

00000041 BA 00000059 R mov edx, OFFSET str1

00000046 E8 00000000 E call WriteString

0000004B return: ret

0000004B 5B \* pop ebx

0000004C C3 \* ret 00000h

0000004D printBlue:

0000004D 66| B8 0081 mov ax,blue + (gray \* 16)

00000051 E8 00000000 E call SetTextColor

00000056 BA 00000059 R mov edx, OFFSET str1

0000005B E8 00000000 E call WriteString

ret

00000060 5B \* pop ebx

00000061 C3 \* ret 00000h

00000062 DisplayTiger endp

end main

\_Microsoft (R) Macro Assembler Version 14.28.29337.0 03/03/21 19:03:00

hw6-1.asm Symbols 2 - 1

Screenshots for problem 1:

When BX is set to 1:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

A picture containing text, screenshot, computer, computer

Description automatically generated

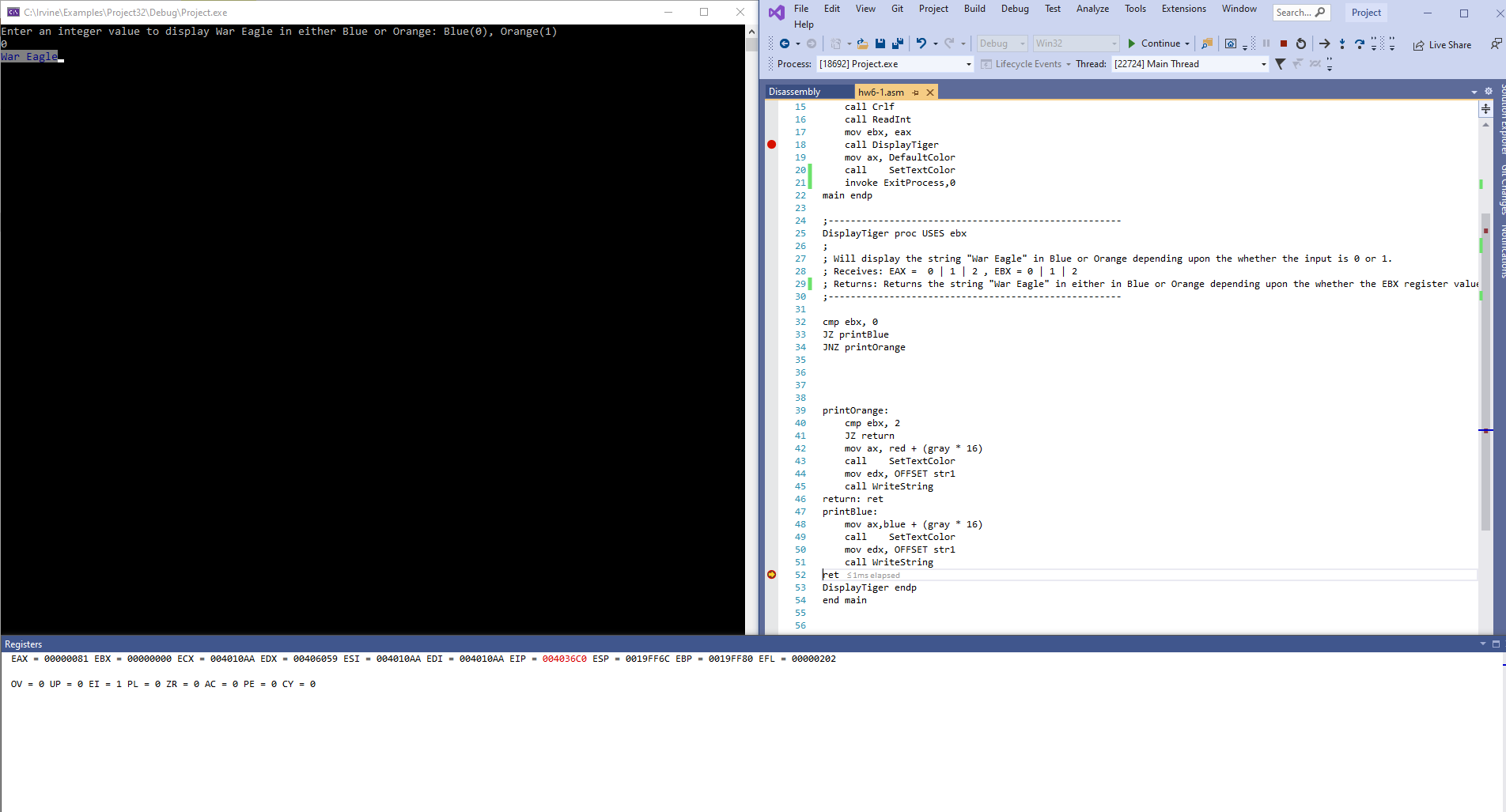
When BX is set to 0:

Graphical user interface, application

Description automatically generated

A picture containing text, screenshot, computer

Description automatically generated

Graphical user interface, application

Description automatically generated

When BX is set to 2:

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

2. [Arrays] Write a program that:

* 1. Prompts the user for integer input 5 times
  2. Stores these inputs in a stack using the Push instruction
  3. After the storing is complete in Step 2, pop the stored values and display them on the screen using WriteInt (notDumpRegs).

Use the following:

.data  
PromptUser BYTE "Please enter a value:", 0

In your submission, please embed the full program (.asm and .lst file) and one screen shot with at least one positive and one negative input value.

List file for hw6-2.asm

hw6-2.asm Page 1 - 1

; hw6-2.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 50 6C 65 61 73 PromptUser BYTE "Please enter a value:", 0

65 20 65 6E 74

65 72 20 61 20

76 61 6C 75 65

3A 00

00000016 48 65 72 65 20 Stackcontent BYTE "Here are the stack values:", 0

61 72 65 20 74

68 65 20 73 74

61 63 6B 20 76

61 6C 75 65 73

3A 00

00000031 00000005 [ inputArray DWORD 5 DUP(?)

00000000

]

00000000 .code

00000000 main proc

00000000 B9 00000005 mov ecx, LENGTHOF inputArray

00000005 BE 00000031 R mov esi, OFFSET inputArray

0000000A L1:

0000000A BA 00000000 R mov edx,OFFSET PromptUser

0000000F E8 00000000 E call WriteString

00000014 E8 00000000 E call Crlf

00000019 E8 00000000 E call ReadInt

0000001E 50 push eax

0000001F E2 E9 loop l1

00000021 E8 00000000 E call Crlf

00000026 BA 00000016 R mov edx, OFFSET Stackcontent

0000002B E8 00000000 E call WriteString

00000030 B9 00000005 mov ecx, LENGTHOF inputArray

00000035 L2:

00000035 58 pop eax

00000036 E8 00000000 E call Crlf

0000003B E8 00000000 E call WriteInt

00000040 E2 F3 loop L2

invoke ExitProcess,0

00000042 6A 00 \* push +000000000h

00000044 E8 00000000 E \* call ExitProcess

00000049 main endp

end main

\_Microsoft (R) Macro Assembler Version 14.28.29337.0 03/03/21 20:37:08

hw6-2.asm Symbols 2 - 1

.asm file for hw6-2.asm

; hw6-2.asm

INCLUDE Irvine32.inc

.data

PromptUser BYTE "Please enter a value:", 0

Stackcontent BYTE "Here are the stack values:", 0

inputArray DWORD 5 DUP(?)

.code

main proc

mov ecx, LENGTHOF inputArray

mov esi, OFFSET inputArray

L1:

mov edx,OFFSET PromptUser

call WriteString

call Crlf

call ReadInt

push eax

loop l1

call Crlf

mov edx, OFFSET Stackcontent

call WriteString

mov ecx, LENGTHOF inputArray

L2:

pop eax

call Crlf

call WriteInt

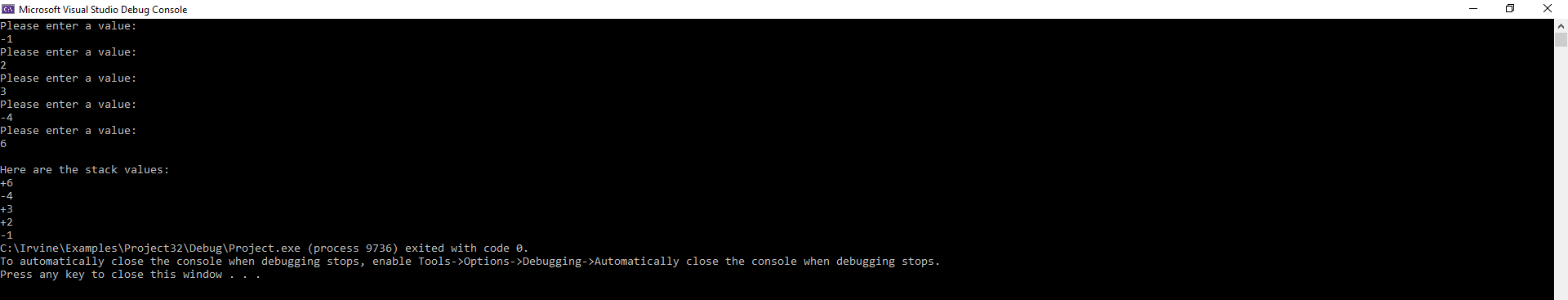
loop L2

invoke ExitProcess,0

main endp

end main

screen shot with at least one positive and one negative input value.



3. [Compares, Procedures] Write a procedure, *Search* which searches the stack for the value that you provide in the register AX and returns its index, assuming the first value is stored in index *0*.   Write a main program that fills the stack with negative values, sets AX and calls *Search* and prints the index at which the value was found.

For example, if the inputs are:  -5, -6, -1, -10, -44, -79

and AX is set in the main program to be -1, then the expected output of your code is:

The target value is -1, and is located at index: 2

In cases where more than one element has the same value, you only have to output one of them. If the value is not found, print 0.

Use the following:

.data  
prompt      BYTE "Please input a value: ", 0  
spacing     BYTE ", ",0;  
String2     BYTE "The target value is,” 0  
String2     BYTE “and is located at index: ",0  
String3 BYTE "Value not found,", 0

In your submission, please embed the full program (.asm and .lst file) and one screen shot showing the values found.  Please test several sets of positive and negative values

asm file

; hw6-3.asm

INCLUDE Irvine32.inc

.data

prompt BYTE "Please input a value: ", 0

spacing BYTE ", ",0

Str2 BYTE "The target value is ", 0

Str3 BYTE "and is located at index: ", 0

Str4 BYTE "Value not found,", 0

target SDWORD ?

firstValue SDWORD ?

.code

main proc

mov ecx, 5

mov esi, 0

input:

mov edx, OFFSET prompt

call WriteString

mov edx, OFFSET spacing

call WriteString

call ReadInt

push eax

loop input

mov ebp, esp

mov edx, OFFSET Str2

call WriteString

call ReadInt

call Search

invoke ExitProcess,0

main endp

;-----------------------------------------------------------------------------------------------------------------------------------;

Search proc USES ebp

;-----------------------------------------------------------------------------------------------------------------------------------;

; Searches the stack for the value that you provide in the register AX and returns its index.

; Receives: AX

; Returns: Returns the index of the value stored in AX, assuming the first value is stored in index 0

;-----------------------------------------------------------------------------------------------------------------------------------;

mov ebx, [ebp + 16]

mov firstValue, ebx

cmp eax, [ebp]

JZ fifthVal

JNZ forthVal

fifthVal:

mov edx, OFFSET Str3

call WriteString

mov eax, 4

call WriteDec

ret

forthVal:

cmp eax, [ebp + 4]

JNZ thirdVal

mov edx, OFFSET Str3

call WriteString

;pop [ebp + 4]

mov eax, 3

call WriteDec

ret

thirdVal:

cmp eax, [ebp + 8]

JNZ secVal

mov edx, OFFSET Str3

call WriteString

mov eax, 2

call WriteDec

ret

secVal:

cmp eax, [ebp + 12]

JNZ fVal

mov edx, OFFSET Str3

call WriteString

mov eax, 1

call WriteDec

ret

fVal:

cmp eax, [ebp + 16]

JNZ ValNotFound

mov edx, OFFSET Str3

call WriteString

mov eax, 0

call WriteDec

ret

ValNotFound:

mov edx, OFFSET Str4

call WriteString

mov eax, -1

call WriteInt

ret

Search endp

end main

list file

; hw6-3.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 50 6C 65 61 73 prompt BYTE "Please input a value: ", 0

65 20 69 6E 70

75 74 20 61 20

76 61 6C 75 65

3A 20 00

00000017 2C 20 00 spacing BYTE ", ",0

0000001A 54 68 65 20 74 Str2 BYTE "The target value is ", 0

61 72 67 65 74

20 76 61 6C 75

65 20 69 73 20

00

0000002F 61 6E 64 20 69 Str3 BYTE "and is located at index: ", 0

73 20 6C 6F 63

61 74 65 64 20

61 74 20 69 6E

64 65 78 3A 20

00

00000049 56 61 6C 75 65 Str4 BYTE "Value not found,", 0

20 6E 6F 74 20

66 6F 75 6E 64

2C 00

0000005A 00000000 target SDWORD ?

0000005E 00000000 firstValue SDWORD ?

00000000 .code

00000000 main proc

00000000 B9 00000005 mov ecx, 5

00000005 BE 00000000 mov esi, 0

0000000A input:

0000000A BA 00000000 R mov edx, OFFSET prompt

0000000F E8 00000000 E call WriteString

00000014 BA 00000017 R mov edx, OFFSET spacing

00000019 E8 00000000 E call WriteString

0000001E E8 00000000 E call ReadInt

00000023 50 push eax

00000024 E2 E4 loop input

00000026 8B EC mov ebp, esp

00000028 BA 0000001A R mov edx, OFFSET Str2

0000002D E8 00000000 E call WriteString

00000032 E8 00000000 E call ReadInt

00000037 E8 00000007 call Search

invoke ExitProcess,0

0000003C 6A 00 \* push +000000000h

0000003E E8 00000000 E \* call ExitProcess

00000043 main endp

;-----------------------------------------------------------------------------------------------------------------------------------;

00000043 Search proc USES ebp

;-----------------------------------------------------------------------------------------------------------------------------------;

; Searches the stack for the value that you provide in the register AX and returns its index.

; Receives: AX

; Returns: Returns the index of the value stored in AX, assuming the first value is stored in index 0

;-----------------------------------------------------------------------------------------------------------------------------------;

00000043 55 \* push ebp

00000044 8B 5D 10 mov ebx, [ebp + 16]

00000047 89 1D 0000005E R mov firstValue, ebx

0000004D 3B 45 00 cmp eax, [ebp]

00000050 74 02 JZ fifthVal

00000052 75 16 JNZ forthVal

00000054 fifthVal:

00000054 BA 0000002F R mov edx, OFFSET Str3

00000059 E8 00000000 E call WriteString

0000005E B8 00000004 mov eax, 4

00000063 E8 00000000 E call WriteDec

ret

00000068 5D \* pop ebp

00000069 C3 \* ret 00000h

0000006A forthVal:

0000006A 3B 45 04 cmp eax, [ebp + 4]

0000006D 75 16 JNZ thirdVal

0000006F BA 0000002F R mov edx, OFFSET Str3

00000074 E8 00000000 E call WriteString

;pop [ebp + 4]

00000079 B8 00000003 mov eax, 3

0000007E E8 00000000 E call WriteDec

ret

00000083 5D \* pop ebp

00000084 C3 \* ret 00000h

00000085 thirdVal:

00000085 3B 45 08 cmp eax, [ebp + 8]

00000088 75 16 JNZ secVal

0000008A BA 0000002F R mov edx, OFFSET Str3

0000008F E8 00000000 E call WriteString

00000094 B8 00000002 mov eax, 2

00000099 E8 00000000 E call WriteDec

ret

0000009E 5D \* pop ebp

0000009F C3 \* ret 00000h

000000A0 secVal:

000000A0 3B 45 0C cmp eax, [ebp + 12]

000000A3 75 16 JNZ fVal

000000A5 BA 0000002F R mov edx, OFFSET Str3

000000AA E8 00000000 E call WriteString

000000AF B8 00000001 mov eax, 1

000000B4 E8 00000000 E call WriteDec

ret

000000B9 5D \* pop ebp

000000BA C3 \* ret 00000h

000000BB fVal:

000000BB 3B 45 10 cmp eax, [ebp + 16]

000000BE 75 16 JNZ ValNotFound

000000C0 BA 0000002F R mov edx, OFFSET Str3

000000C5 E8 00000000 E call WriteString

000000CA B8 00000000 mov eax, 0

000000CF E8 00000000 E call WriteDec

ret

000000D4 5D \* pop ebp

000000D5 C3 \* ret 00000h

000000D6 ValNotFound:

000000D6 BA 00000049 R mov edx, OFFSET Str4

000000DB E8 00000000 E call WriteString

000000E0 B8 FFFFFFFF mov eax, -1

000000E5 E8 00000000 E call WriteInt

ret

000000EA 5D \* pop ebp

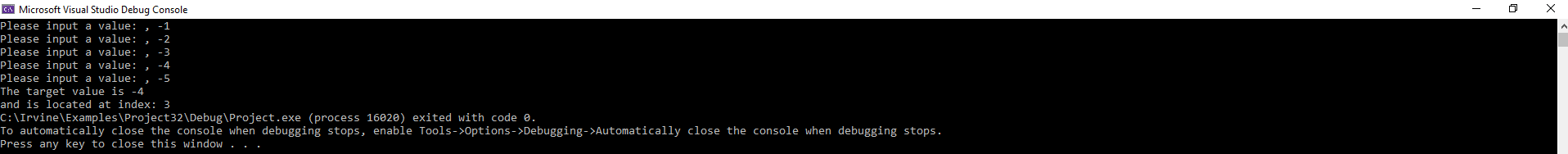
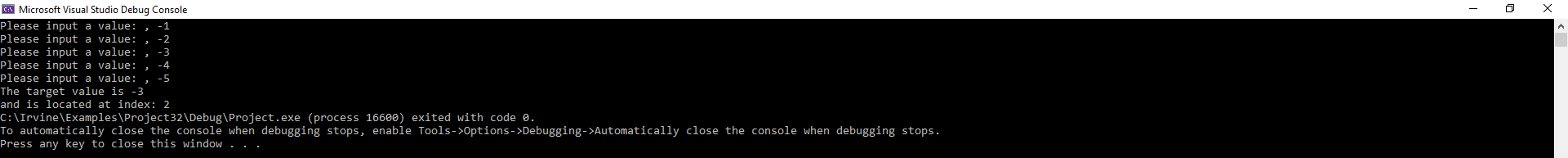
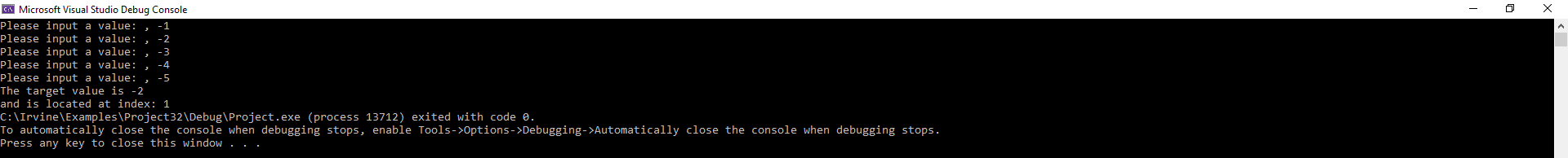
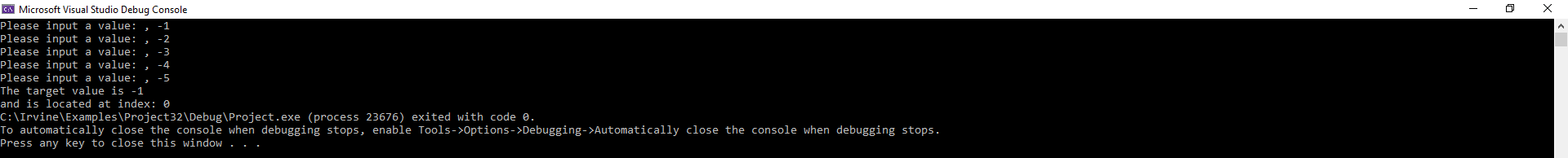
000000EB C3 \* ret 00000h

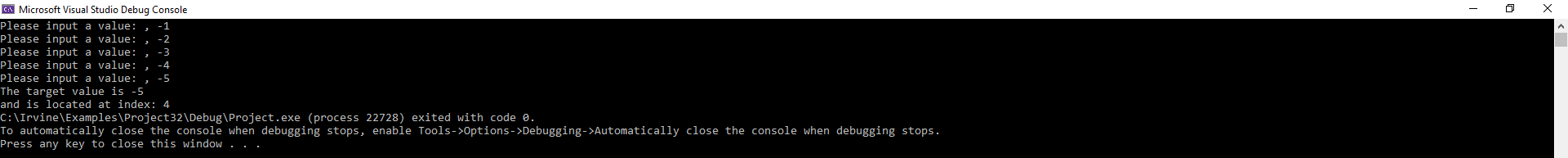
000000EC Search endp

end main

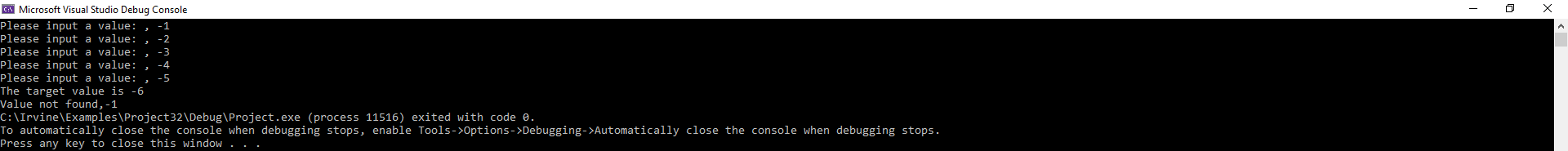
\_Microsoft (R) Macro Assembler Version 14.28.29337.0 03/03/21 20:49:12

hw6-3.asm Symbols 2 – 1





Value not found:



where more than one element has the same value, and only one of them is outputted

output shows the index of the last instance of the repeated element -5 in the screenshot below:

