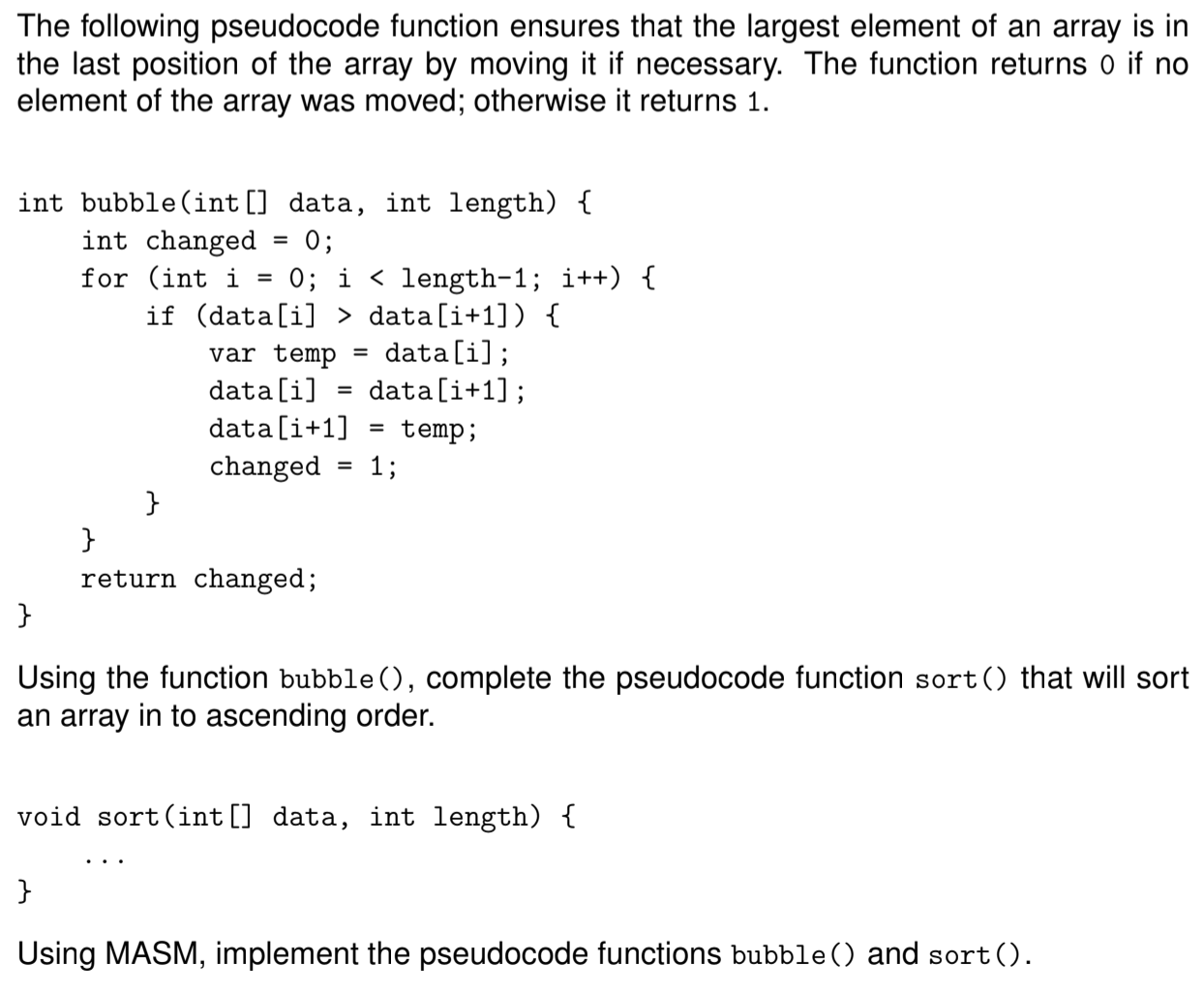
2018 Q6



**sort.asm**

EXTERNDEF bubble:near

.data

M = 10

givenArray DWORD 10,6,1,7,9,3,4,5,2,8

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke version

invoke writeArray, addr Array, M ; write the original array

send\_array:

push offset givenArray ; push the array

push M ; push the length of array

call bubble ; call bubble

cmp eax, 1 ; compare toggle to 1

je send\_array ; if greater than 0 repeat loop

finish:

invoke writeArray, addr givenArray, M ; print the sorted array

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**bubble.asm**

EXTERNDEF bubble:near

.code

bubble:

push ebp ; know where to come back to after function

mov ebp, esp ; put the pointers in the same place

sub esp, 16 ; making space for index i = [ebp-4], etc

mov eax, 0 ; make eax = 0

mov [ebp-4], eax ; index = 0

mov [ebp-8], eax ; changed = 0

mov [ebp-12], eax ; this will be our toggle (1 or 0)

inc eax ; eax += 1

mov [ebp-16], eax ; j index (i+1) = 1

go\_through:

mov eax, [ebp-4] ; put i into eax

mov ebx, [ebp-16] ; put j into ebx

cmp ebx, [ebp+8] ; compare j to the length of array

je count\_changes ; if they are equal we count changes and get ready to leave

shl eax, 2 ; i\*4

add eax, [ebp+12] ; add i index to array address to get A[i]

mov ecx, [eax] ; move the value in ith position to ecx

shl ebx, 2 ; j\*4

add ebx, [ebp+12] ; add j index to array address to get A[j]

mov edx, [ebx] ; move the value in the jth position to edx

cmp ecx, edx ; compare ecx to edx

ja swap\_em ; if ecx is greater we need to swap

jmp reset ; else we move on

swap\_em: ; this swaps the elements

mov [eax], edx ; putting the value in edx in the memory location stored in [eax]

mov [ebx], ecx ; putting the value in ecx in the memory location stored in [ebx]

inc DWORD ptr[ebp-8] ; increment the changes counter

reset:

inc DWORD ptr[ebp-4] ; increment i

inc DWORD ptr[ebp-16] ; increment j

jmp go\_through ; go back through again

count\_changes:

mov eax, [ebp-8] ; move the number of changes to eax

cmp eax, 0 ; if changes have taken place eax is greater than 0

je no\_change ; if zero go to no\_change, else drop into yes\_change

jmp yes\_change

yes\_change:

inc DWORD ptr[ebp-12] ; toggle += 1 - a change has been made

jmp go\_home ; jump to go back to main

no\_change:

mov eax, 0 ; eax = 0

mov [ebp-12], eax ; toggle = 0 because no changes

go\_home:

mov eax, [ebp-12] ; move the toggle into eax to send it back

mov esp, ebp ; standard return to reset stack pointer

pop ebp ; standard pop to take off ebp pointer from stack

ret 8 ; return by (number of values pushed in main)\*4

end

2017 Q6

; Using recursion, write a pseudocode function largest to find the

; largest element in a non-empty array of integers.

;

; Using MASM, implement largest.

;

; **main.asm:**

;

; Data & Code

;

EXTERNDEF largest:near

.data

N = 6 ; length of array

givenArray DWORD 10, 20, 79, 44, 36, 12

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke version

mov ebx, N ; put length of array into ebx

dec ebx ; get index number of last position in array

sal ebx, 2 ; ebx times 4 (number of bytes in each)

add ebx, offset givenArray ; add ebx to offset givenArray (offset givenArray is always index 0)

push [ebx] ; last number in array

push N ; length of array

push offset givenArray ; first element in array

call largest

finish:

invoke writeInteger, ebx ; our previous largest is here: print it out

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**largest.asm**

EXTERNDEF largest: near

.data

i DWORD 1

.code

largest:

push ebp

mov ebp, esp

mov ebx, [ebp+16] ; moving N into ebx

mov ecx, i ; move i into register ecx

mov eax, [ebp+12] ; taking previous biggest value from stack and putting it in eax

cmp ecx, eax ; compare i with N

je finish ; when i is len(array), finish

dec eax ; N minus one

sub eax, ecx ; N minus i, answer stored in eax

sal eax, 2 ; multiply answer by 4, to get next pointer position

add eax, [ebp+8] ; add new position to offset array

cmp [eax], ebx ; compare previous biggest to what is in eax

jg switcheroo ; if its larger, we need to swap it

jmp pushing ; if its not larger, dont swap, go back into loop

switcheroo:

mov ebx, [eax] ; move value into previous biggest

pushing:

inc i

push ebx

push [ebp+12]

push [ebp+8]

call largest

finish:

mov esp, ebp

pop ebp

ret 12

end