

Topic	Practical Assignment 3 Cover Sheet
Assignment Type	<input checked="" type="checkbox"/> Assessed <input type="checkbox"/> Non-assessed <input checked="" type="checkbox"/> Individual <input type="checkbox"/> Group
Module	CSE101 Computer Systems
Due Date	November 3 rd , 2017 (Friday)
Student ID	1719247
Student Name	Christopher Champion
Submission Date	Nov 1, 2017

Declaration on Plagiarism and Collusion

I have read and understood the definitions of plagiarism and collusions as described in the University's Code of Practice on Assessment. As such, I certify the work presented in this report/assignment has been written solely by me and in my own words (except where references and acknowledgments are clearly defined). I agree to accept disciplinary actions should I be caught with the serious offence of plagiarism and/or collusion.

Signature:  _____

For Academic Use	Date Received	No. of Days Late	Penalty

Program Listing

```
// Practical3Champion171247Final.cpp : Defines the entry point for the console application.
#include "stdafx.h"

int main(int argc, _TCHAR* argv[])
{
    char colonMsg[] = ": ";
    char commaMsg[] = ", ";
    char format[] = "%d"; // format string for the scanf function
    char arraySizeMsg[] = "Select total number of positive integers (between 2-5): ";
    char infoMsg[] = "\n*****\n\nYou may enter a negative number to exit input mode early.\n\n*****";
    char enterPosIntMsg[] = "\nEnter positive integer ";
    char invalidInputMsg[] = "\nYou entered an invalid number.\n\n";
    char minTwoNumbersMsg[] = "\nYou must enter at least two positive integers.\n\n";
    char arrayValuesMsg[] = "\n\nYou entered the following values: ";
    char bubbleSortMsg[] = "\n\nYour integers from lowest to highest are: ";
    char arraySumMsg[] = "\n\nThe total amount is: ";
    char loopCountMsg[] = "\n*****\n\nProgram terminates and has looped ";
    char timesMsg[] = " times.\n\n*****";
    char programEndMsg[] = "\n\nType in any value and press RETURN to finish: ";
    int counter;
    int numberArraySum;
    int minLoopVal = 2;
    int maxLoopVal = 5;
    int numberArray[5];
    int numberArraySize = 0;
    int programEndInput;

    _asm
    {
        // 1. GET COUNTER VALUE

        lea     eax, arraySizeMsg           ; LOAD pointer to arraySizeMsg[0] into EAX
        push   eax
        getCounter :    call    printf       ; LOOP: print message and read counter in
                        lea     eax, counter ; LOAD pointer to counter into EAX
                        push   eax
                        lea     eax, format  ; LOAD pointer to format for scanf parameter
                        push   eax
                        call    scanf_s; CALL C scan function to read in counter
                        add     esp, 8       ; ADD 8 bytes for counter and format push

        mov     eax, counter                ; Compare input with permissible max/min value
        mov     ebx, maxLoopVal             ; MOVE max loop value of 5 into ebx for comparing
        cmp     eax, ebx                    ; Compare the two values and jump if ebx > eax
        jg      invalidInput                ; Jumps or falls through
        mov     ebx, minLoopVal             ; MOVE 2 into ebx for comparing
        cmp     eax, ebx                    ; Compare and jump if eax < ebx
        jl      invalidInput                ; Jumps or falls through

        jmp     leaveCounterLoop            ; If counter is valid (2 to 5) leave loop

        invalidInput : lea     eax, invalidInputMsg
                        push   eax
                        call    printf
                        add     esp, 4
                        jmp     getCounter    ; Print invalid input, jump to getCounter

        leaveCounterLoop : add esp, 4        ; ADD 4 bytes to stack pointer for push to eax

        lea     eax, infoMsg                ; Print exit loop info message
        push   eax
        call    printf
        add     esp, 4
    }
}
```



```
// 2. GET VALUES FROM USER TO STORE IN numberArray

mov     ebx, counter                ; MOVE counter value into ebx register
lea     esi, numberArray
jmp     printEnterMsg

negIntEntered : dec     numberArraySize    ; Jumped to from printEnterMsg on negative int input
              mov     eax, numberArraySize
              cmp     eax, minLoopVal
              jl      minTwoNumbers        ; Inform user of min. pos. int requirement
              jge     countLoops

minTwoNumbers : lea     eax, minTwoNumbersMsg ; Print minimum 2 positive ints message
              push    eax
              call    printf
              add     esp, 4

printEnterMsg : lea     eax, enterPosIntMsg  ; LOOP: Enter n(=counter) positive integers
              push    eax
              call    printf
              add     esp, 4
              inc     numberArraySize      ; INC num of ints entered (init. with 0)

              mov     eax, numberArraySize ; Print 'n' in "Enter Integer n"
              push    eax
              lea     eax, format
              push    eax
              call    printf
              add     esp, 8

              lea     eax, colonMsg ; Print ": "
              push    eax
              call    printf
              add     esp, 4

              push    esi                ; PUSH numberArray to the stack
              lea     eax, format        ; CALL scanf to read user input in
              push    eax
              call    scanf_s
              add     esp, 8

              mov     eax, [esi]         ; MOVE user input value into EAX register
              cmp     eax, 0             ; Compare user input to 0
              jl      negIntEntered      ; Jump out of loop if user input < 0

              add     esi, 4             ; Increment esi by 4 to increase array index
              dec     ebx                ; Decrement counter value stored in EBX
              jnz     printEnterMsg      ; Loop if counter != 0, else fall through

countLoops :  lea     eax, loopCountMsg ; Print the loop count message
              push    eax
              call    printf
              add     esp, 4

              mov     eax, numberArraySize ; Array size == number of loop counts
              push    eax
              lea     eax, format        ; Parse int to string
              push    eax
              call    printf
              add     esp, 8             ; Print number of loops

              lea     eax, timesMsg ; Print " times."
              push    eax
              call    printf
              add     esp, 4

// 3. PRINT UNSORTED NUMBER ARRAY

lea     eax, arrayValuesMsg            ; Print arrayValuesMsg
```



```

push    eax
call    printf
add     esp, 4

lea     esi, numberArray      ; Point to numberArray[0]
mov     ecx, numberArraySize  ; Use array size as loop counter
dec     ecx                  ; Decrement array size by 1
                                ; (last value to be printed separately)
                                ; PUSH ecx onto stack to restore after printf call
loopNumArr : push    ecx
              mov     eax, [esi]
              add     esi, 4      ; Point to the next element in numberArray
              push    eax
              lea     eax, format
              push    eax
              call    printf      ; Print the value at current array index
              add     esp, 8

              lea     eax, commaMsg ; Print comma separator
              push    eax
              call    printf
              add     esp, 4
              pop     ecx          ; POP ecx to retrieve value
              loop    loopNumArr   ; Decrement ecx by 1, loop if not 0

mov     eax, [esi]             ; ESI curr. points to the last value in array
push    eax
lea     eax, format
push    eax
call    printf
add     esp, 8

// 4. IMPLEMENT BUBBLE SORT

mov     ecx, numberArraySize  ; Outer loop counter = array size - 1
dec     ecx                  ; Decrement array size by 1

loop1 : push    ecx            ; Save outer loop count
        lea     esi, numberArray ; Point to the first value in numberArray
loop2 : mov     eax, [esi]      ; Move array value at index into eax
        cmp     [esi + 4], eax ; Compare value at index+1 with current value
        jge     loop3          ; if [esi] <= [edi], skip
        xchg    eax, [esi + 4] ; else exchange the pair
        mov     [esi], eax
loop3 : add     esi, 4          ; MOVE both pointers forward
        loop    loop2          ; Inner loop
        pop     ecx            ; Leave outer loop count
        loop    loop1          ; else repeat outer loop

// 5. PRINT BUBBLE SORTED NUMBER ARRAY

lea     eax, bubbleSortMsg     ; Print bubble sort message
push    eax
call    printf
add     esp, 4

lea     esi, numberArray      ; Point to numberArray[0]
mov     ecx, numberArraySize  ; Decrement array size by 1
dec     ecx                  ; (last value to be printed separately)
                                ; PUSH ecx to restore after printf call
loopBubbleArr : push    ecx
                mov     eax, [esi]
                add     esi, 4      ; Point to the next element in numberArray
                push    eax
                lea     eax, format
                push    eax
                call    printf
                add     esp, 8

                lea     eax, commaMsg ; Print comma separator
                push    eax

```

```

        call    printf
        add     esp, 4

        pop     ecx          ; POP ecx to retrieve value
        loop    loopBubbleArr ; Decrement ecx by 1, loop if not 0

mov     eax, [esi]          ; esi currently points to the last value in array
push    eax
lea     eax, format
push    eax
call    printf
add     esp, 8

// 6. ADD ARRAY VALUES AND PRINT numberArraySum

lea     esi, numberArray    ; Point to numberArray[0]
mov     ebx, numberArraySize ; Use array size for loop counter
mov     eax, 0              ; Store sum of array values in eax (init to 0)

addVals :
add     eax, [esi]          ; Sum numberArray[0] + eax (init with 0)
add     esi, 4              ; Point to the next element in numberArray
dec     ebx                 ; Decrement numberArraySize by 1
jnz     addVals             ; Loop if numberArraySize > 0

mov     numberArraySum, eax ; Move eax value into numberArraySum

lea     eax, arraySumMsg     ; Load arraySumMsg pointer into eax reg
push    eax                 ; Push to the stack for printing
call    printf              ; Call C method to print arraySumMsg
add     esp, 4              ; Add 4 bytes to stack pointer for string push

push    numberArraySum      ; PUSH numberArraySum value to the stack
lea     eax, format
push    eax
call    printf              ; CALL C method to print numberArraySum
add     esp, 8              ; ADD 8 bytes to stack pointer

// 7. PRINT programEndMsg AND GET USER INPUT TO QUIT
finish :
lea     eax, programEndMsg   ; Point to endMessage[] first char
push    eax                 ; Push eax to the stack
call    printf              ; Print endMessage
add     esp, 4              ; Add 4 bytes to stack pointer for eax push

lea     eax, programEndInput ; Point to programEndInput integer
push    eax                 ; PUSH eax to the stack
lea     eax, format         ; Load effective address of format char[]
push    eax                 ; Push eax to the stack
call    scanf_s             ; Call scanf_s function for user input
add     esp, 8              ; Add 8 bytes to the stack pointer
}
return 0;
}

```