Topic	Practical Assignment 4 Cover Sheet
Assignment Type	☒ Assessed ☐ Non-assessed ☒ Individual ☐ Group
Module	CSE101 Computer Systems
Due Date	December 6th, 2017 (Wednesday)
Student ID	1719247
Student Name	Christopher Michael Champion
Submission Date	Dec 06, 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: _____

	Date Received	**No. of Days Late**	**Penalty**
For Academic Use			

Program Listing

```
// Practical4Champion1719247.cpp : Defines the entry point for the console
application.
#include "stdafx.h"
#include "string.h"
#define MAX_SZ 500  // Arrays max length of 500
#define MAX_CHAR 50 // User can enter max 50 chars


int main(int argc, _TCHAR* argv[])
{
      char format[] = "%d";   // Format for converting integer to string
                              // decimal representation
      char nameInputMsg[] = "Please enter your name (1-50
                             characters): ";
      char nameReversedMsg[] = "\nYour name in reverse is: ";
      char nameLengthMsg[] = "\n\nThe length of your name is ";
      char nullInputMsg[] = "\nPlease enter at least one character: ";
      char longInputMsg[] = "\nExceeded 50 character limit. Try a shorter
                             name: ";
      char programEndMsg[] = "\n\nEnd of program.\n";

      int nameLength = 0;

      char inputArray[MAX_SZ];      // Array to store user input (size = 500)
      char reverseArray[MAX_SZ] = ""; // Array to store inputArray values in
                                      // reverse (size = 500)
      memset(reverseArray, 0, sizeof(reverseArray));
                                    // Initialize reverseArray with zeros


      _asm

      {

      // 1. PROMPT USER TO INPUT NAME

      lea   eax, nameInputMsg ; Load pointer to nameInputMsg[] into eax
      push  eax               ; Push eax to the stack
      call  printf            ; C funtion writes formatted string to stdout
      add   esp, 4            ; Clear the stack


      // 2. GET USER INPUT, STORE IN inputArray (ESI)


userInput:
      lea   esi, inputArray   ; Load pointer to inputArray[0] into esi
      push  esi               ; Push esi to the stack
      call  gets_s            ; C function reads line from stdin, stores each
                              ; char in inputArray[]
```

```
      call  strlen            ; C function stores length of inputArray[]
                              ; (excluding termination null char) in eax
      mov   nameLength, eax   ; Move int value of inputArray[] length from
                              ; eax into nameLength int var
      add   esp, 4            ; Clear the stack


      // 3. CHECK INPUT VALIDITY


      cmp   nameLength, 0     ; Compare length of inputArray[] to 0
      jz    nullInput         ; Jump to nullInput if inputArray[] length == 0

      cmp   nameLength, MAX_CHAR    ; Compare length of inputArray[] to 50
      jg    longInput         ; Jump to longInput if inputArray[] length > 50

      jmp   reverseInput      ; Jump to reverseInput if user input is valid,
                              ; else:

nullInput :                   ; User must enter at least one value
      lea   eax, nullInputMsg
      push  eax
      call  printf
      add   esp, 4
      jmp   userInput         ; Jump back to userInput

longInput :
      lea   eax, longInputMsg ; Inform user they entered too many characters
      push  eax
      call  printf
      add   esp, 4
      jmp   userInput         ; Jump back to userInput


      // 4. REVERSE INPUT


reverseInput :
      mov   ecx, nameLength   ; Move length of inputArray[] into ecx register
      mov   esi, 0            ; Move 0 into esi stack index register

pushIn :
      movzx eax, inputArray[esi]   ; Move contents of inputArray[i] into eax,
                                   ; zero out higher bytes
      push  eax               ; Push inputArray[i] to the stack
      inc   esi              ; Increment stack pointer by one (char = 1 byte)
      loop  pushIn            ; Decrement ecx by 1, loop until ecx == 0

mov         ecx, nameLength   ; Move length of inputArray[] into ecx register
mov         esi, 0            ; Move 0 into esi stack index register

popOut :
      pop   eax               ; Get char value from top of stack, store in
                              ; eax then clear it from stack
      mov   reverseArray[esi], al   ; Move char value from al (= eax low 8
```

```
                                ; bit register) into reverseArray[i]
inc   esi               ; Increment stack pointer by one (char = 1 byte)
loop  popOut             ; Decrement ecx by 1, loop until ecx == 0


// 5. PRINT REVERSE ARRAY


lea   eax, nameReversedMsg    ; Load pointer to nameReversedMsg[] into
                              ; eax register
push  eax                     ; Push eax to the stack
call  printf                  ; C function writes formatted string to
                              ; stdout
add   esp, 4                  ; Clear the stack

lea   eax, reverseArray       ; Load pointer to reverseArray[] into eax
push  eax                     ; Push eax to the stack
call  printf                  ; C function writes formatted string to
                              ; stdout
add   esp, 4                  ; Clear the stack


// 6. PRINT NAME STRING LENGTH


lea   eax, nameLengthMsg      ; Load pointer to nameLengthMsg[] into
                              ; eax register
push  eax                     ; Push eax to the stack
call  printf                  ; C function writes formatted string to
                              ; stdout
add   esp, 4                  ; Clear the stack


push  nameLength              ; Push length of inputArray[] into eax
lea   eax, format             ; Load pointer to format[] string into
                              ; eax register
push  eax                     ; Push eax to the stack

call  printf                  ; C function writes formatted string to
                              ; stdout
add   esp, 8                  ; Clear the stack


// 7. PRINT END MESSAGE


lea   eax, programEndMsg      ; Load pointer to programEndMessage[]
                              ; into eax register
push  eax                     ; Push eax to the stack
call  printf                  ; C function writes formatted string to
                              ; stdout
add   esp, 4                  ; Clear the stack

}
```

```
        return 0;
}
```