| | |
|---|---|
| **Topic** | Practical Assignment 3 Cover Sheet |
| **Assignment Type** | ☒ Assessed ☐ Non-assessed<br>☒ Individual ☐ Group |
| **Module** | CSE101 Computer Systems |
| **Due Date** | November 3rd, 2017 (Friday) |
| **Student ID** | 1614649 |
| **Student Name** | Kai-Yu Lu |
| **Submission Date** | 2017.11.02 |

**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: _______________Kai-Yu Lu__________________

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

## Program Listing

```
int main()
{
        int input_arr[5];                   //The number of integers that users can input.
        int t = 0;                          // The times that user want to run the loop.
        int l = 0;                          //The times that add_loop ran to avoid getting the
                                             opposite order of the loop that program runs,

        int total = 0;
        char loopCounter_input[] = "Select total number of positive integers (between 2-5): ";
        char enter_input[] = "Enter positive integer %d: ";
        char proRunLoop[] = "Program terminates and has looped %d times.\n";
        char order_input[] = "Your integers from lowest to highest is ";
        char total_input[] = "\nThe total amount is %d";
        char end_sentence[] = "\nPress any button to continue...";
        char format[] = "%d";                //To print a signed decimal number
        char sep_input[] = ", ";             //To separate the numbers have been printed.

        __asm {
        t_loop:
                lea eax, loopCounter_input;   //Load address of the string  'loopCounter_input'
                                               into eax.
                push eax;                     //Address of string, stack parameter call.
                call printf;                  //Use library code subroutine.
                add esp, 4;                   //Clean 4 byte parameter off stack.

        lea eax, t;                           //Load address of the string 't' into eax.
        push eax;                             //Address of string, stack parameter call.
        lea eax, format;                      //Load address of the string 'format' into eax.
        push eax;                             //Address of string, stack parameter call.
    call scanf;                               //It will take two parameters from the stack;
                                                scanf(%d,& t).
                                                Here I want to realize the funtion of scanning
                                                the running times of loop that user want.
        add esp, 8;                           //Clean 8 byte parameter off stack.

        mov eax, t;                           //Load the variable "t" into eax.
        cmp eax, 5;                           //Compare eax with the number 5.
        jg t_loop;                            //Jump to "t_loop" if eax (inside the number is
                                                t)>5.
        cmp eax, 2;                           //Compare eax with the number 2.
        jl t_loop;                            //Jump to "t_loop" if eax (inside the number is t) <
                                                2.
                                              //This logic is like "if" loop and set the input
                                                number should be larger than 2 and less than 5.
        mov ebx, 0;                           //Load the constant "0" into ebx. It aims to use ebx
                                                in the loop.
        mov ecx, t;                           //Initialize loop counter which is expressed by
                                                    variable "t".
        add_loop:
```

```
        push ecx;                    //Loop count index saved on stack.

        inc l;                       //l means how many times this loop has run. Everytime
                                       this loop runs, l will add 1 to realize the
                                       function of the loop conut.
        mov eax, l;                  //Load the constant "0" into eax.
        push eax;                    //Address of string, stack parameter call.


        lea  eax, enter_input;       //Load address of the string 'enter_input' into eax.
        push eax;                    //Address of string, stack parameter call.
        call printf;                 //It will take two parameters from the stack;
                                       printf(%d, & l).
        add esp, 8;                  //Clean 8 byte parameter off stack.

        lea eax, input_arr[ebx];     //Address of the array (its 0th element) is saved in
                                       ebx. Here I want to realize the function of
                                       scanning the number that user input.
        push eax;                    //Address of string, stack parameter call.
        lea eax, format;             //Load address of the string 'format' into eax.
        push eax;                    //Address of string, stack parameter call.
        call scanf;                  //It will take two parameters from the stack;
                                       scanf(%d,& [ebx]).
        add esp, 8;                  //Clean 8 byte parameter off stack.

        mov eax, input_arr[ebx];     //Let the number that user input be put into eax to
                                       be printed.
        cmp eax, 0;                  //Compare the input number with 0.
        jg  minusL;                  //If the input number > 0, jump to "minusL".
        add esp, 4;                  //Clean 4 byte parameter off stack.
        dec l;                       //If the input number < 0, l minus 1. It aims to
                                       realize the function that if the input number was
                                       neagative, the real times that the loop runs
                                       will reduce one. In addition, it is in the loop,
                                       thus times of loop will execute this operation
                                       automatically.
        jmp Bubble;                  //Next, having finished counting the running times of
                                       the loop, it will jump to "Bubble" to sort the
                                       input numbers.

minusL:                              //This loop is used to continue running if the input
                                       number is positive.
        add ebx, 4;                  //All input numbers' position will move forward once.

        add total, eax;              //Original total  is 0. If run to this line, the
                                       total will be added by the number in eax, which
                                       realizes the funtion of summing up.
        push total;                  //Address of string, stack parameter call.
        add esp, 4;                  //Clean 4 byte parameter off stack.
```

```
                 pop ecx;                    //Restore loop counter ready for test.
                 loop add_loop;              //Continue running "add_loop".

Bubble:                                      //This loop is used to sort the input numbers.
                 mov eax, 1;                 //Now, thhis 1 expresses the running times of the
                                               loop and which does not contain the times when the
                                               input number is negative.

                 push eax;                   //Address of string, stack parameter call.
                 lea eax, proRunLoop;        //Load address of the string 'proRunLoop' into eax.
                 push eax;                   //Address of string, stack parameter call.
                 call printf;                //It will take two parameters from the stack;
                                               printf(%d, & 1)
                add esp, 8;                  //Clean 8 byte parameter off stack.

                lea eax, order_input;        //Load address of the string 'order_input' into eax.
                push eax;                    //Address of string, stack parameter call.
                call printf;                 //Use library code subroutine.
                add esp, 4;                  //Clean 4 byte parameter off stack.

                mov ecx, 1;                  //The begin of Bubble Sort.
                dec ecx;                     //Decrement count by 1.
                L1 : push ecx;               //Save outer loop count.
                lea  esi, input_arr;         //Point to first value.
                L2 : mov eax, [esi];         //Get array value.
                cmp[esi + 4], eax;           //Compare the previous input number and later number
                                               inputted.
                jle L3;                      //If [esi] <= [edi], jump to L3.
                xchg eax, [esi + 4];         //Else exchange the pair.
                mov[esi], eax;               //Store the value in eax to [esi] because eax will be
                                               used later.
                L3 : add  esi, 4;            //Move both pointers forward.
                loop L2;                     //Inner loop.

                mov eax, [esi];              //Now, the input number is stored in to eax and wait
                                               to be printed.
                push eax;                    //Address of string, stack parameter call.
                lea eax, format;             //Load address of the string 'format' into eax.
                push eax;                    //Address of string, stack parameter call.
                call printf;                 //It will take two parameters from the stack;
                                               printf(%d, & [esi]).
                add esp, 8;                  //Clean 8 byte parameter off stack.

                lea eax, sep_input;          //Load address of the string 'sep_input' into eax.
                                               Here why we use it is to separate the numbers have
                                               been printed.

                 push eax;                   //Address of string, stack parameter call.
                 call printf;                //Use library code subroutine..
                 add esp, 4;                 //Clean 4 n\ byte parameter off stack.
```

```
            pop  ecx;                     //Retrieve outer loop count.
            loop L1;                      //else repeat outer loop.

            mov eax, input_arr[0];        //The input number has been stored in array and wait
                                            to be printed.
            push eax;                     //Address of string, stack parameter call.
            lea eax, format;              //Load address of the string 'format' into eax.
            push eax;                     //Address of string, stack parameter call.
            call printf;                  //It will take two parameters from the stack;
                                            printf(%d, & input_arr[0]).
            add esp, 8;                   //Clean 8 byte parameter off stack.

            mov eax, total;               //Now, the sum of the input numbers without negative
                                            number have beed added and it is expressed by
                                            "total".
            push eax;                     //Address of string, stack parameter call.
            lea eax, total_input;         //Load address of the string  'total_input' into eax.
            push eax;                     //Address of string, stack parameter call.
            call printf;                  //It will take two parameters from the stack;
                                            printf(%d, & total).
            add esp, 8;                   //Clean 8 byte parameter off stack.


            lea eax, end_sentence;        /*Load address of the string  'end_sentence' into
                                             eax. Here because the requirement of the question
                                             which is " When the program terminates, print
                                              out an exit and number of loops message."*/
            push eax;                     //Address of string, stack parameter call.
            call printf;                  //Use library code subroutine.
            add esp, 4;                   //Clean 4 byte parameter off stack.
            call getchar;                 //End the program.
            call getchar;
        }
    return 0;
}
```