|  |  |
| --- | --- |
| **Topic** | Practical Assignment 4 Cover Sheet |
| **Assignment Type** | 🗷 Assessed 🞏 Non-assessed  🗷 Individual 🞏 Group |
| **Module** | CSE101 Computer Systems |
| **Due Date** | December 12th, 2018 (Wednesday) |
| **Student ID** | 1717576 |
| **Student Name** | Minhao Jin |
| **Submission Date** | December 12th, 2018 (Wednesday) |

**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**

*#include "stdafx.h"*

*#include "stdlib.h"*

*#include <string>*

*#include <iostream>*

*#include <stdio.h>*

*#include <cmath>*

*using namespace std;*

*int main()*

*{*

*char enter\_looptime[] = "Enter number of students (between 3-10):";*

*char student\_ID\_format[] = "Enter Student ID [%d]: ";*

*char student\_ID\_warning[] = "Warning: Invalid Student ID. Must be between 18000 to 18999.\n";*

*char student\_name\_format[] = "Enter Student Name [%d]: ";*

*char student\_name\_warning[] = "Warning: Invalid Student Name. Must be 10 characters or less.\n";*

*char Grade\_format[] = "Enter Grade [%d]: ";*

*char grade\_warning[] = "Warning: Invalid Grade. Must be between 0 to 100.\n";*

*char result[] = "List of Student IDs, Student Names and their Grades:\n";*

*char fail\_format[] = "Number of students who failed is %d\n";*

*char mean[] = "Mean is %d";*

*char deviation\_format[] = "Standard deviation is %.4f\n";*

*char end[] = "Program ends.\n";*

*char int\_format[] = "%d";*

*char double\_format[] = "%lf";*

*char string\_format[] = "\r%[^\n]";*

*char int\_tab[] = "%d\t";*

*char tab\_int[] = "\t%d";*

*char newline[] = "\n";*

*char order[] = "%d. ";*

*char fraction1[] = ".%d\n";*

*int student\_ID[10];*

*memset(student\_ID, 0, sizeof(student\_ID));*

*int ID\_address = -4;*

*char student\_name[10][30];*

*memset(student\_name, 0, sizeof(student\_name));*

*int grade[10];*

*memset(grade, 0, sizeof(grade));*

*int loopCounter = 0;*

*int current\_loop = 1;*

*int sum = 0;*

*int average = 0;*

*int square\_sum = 0;*

*int sum\_square = 1;*

*int dividend=0;*

*float divisor = 1;*

*float deviation;*

*int fraction\_part = 0;*

*int fail = 0;*

*\_asm {*

*looptimes: //Fuction: to ensure looptimes which the user input is greater than 3 and less than 10*

*lea eax, enter\_looptime;*

*push eax;*

*call printf; //Print "Enter number of students (between 3-10):" on the screen*

*add esp, 4;*

*lea eax, loopCounter;*

*push eax;*

*lea eax, int\_format;*

*push eax;*

*call scanf\_s; //Input the times for loop*

*add esp, 8;*

*mov eax, loopCounter;*

*cmp eax, 3; //Judge whether the valuable named "loopCounter" is less than 3 or not*

*jl looptimes; //Jump to "looptimes" if "loopCounter" is less than 3 for another valid input*

*cmp eax, 10; //Judge whether the valuable named "loopCounter" is greater than 10 or not*

*jg looptimes; //Jump to "looptimes" if "loopCounter" is greater than 10 for another valid input*

*mov ecx, loopCounter; //Save the value of "loopCounter" in ecx*

*lea edi, student\_name; //Save the head address of "student\_name" in edi*

*input\_loop: //The input loop starts here!*

*push ecx; //Protect the value of ecx*

*add ID\_address, 4; //Let the "ID\_address" be 0 at first, add 4 for each loop*

*mov esi, ID\_address; //Save the value of "ID\_address" in esi*

*jmp input\_student\_ID; //Jump to "input\_student\_ID"*

*add\_newline: //Fuction: add a newline between two students' input*

*lea eax, newline;*

*push eax;*

*call printf;*

*add esp, 4;*

*inc current\_loop; //To record the current loop's number in current\_loop*

*pop ecx; //ecx minus 1 for each loop*

*loop input\_loop; //Loop ends*

*jmp sort; //Jump to "sort"*

*//===========================ID==============================*

*input\_student\_ID: //Fuction: to ensure that all the input student IDs are valid*

*mov eax, current\_loop; //Save the value of current\_loop in eax*

*push eax;*

*lea eax, student\_ID\_format;*

*push eax;*

*call printf; // Print "Enter Student ID [current\_loop]:" on the screen*

*add esp, 8;*

*lea eax, student\_ID[esi]; //Save the current address of student\_ID in eax*

*push eax;*

*lea eax, int\_format;*

*push eax;*

*call scanf\_s; //Input the student ID*

*add esp, 8;*

*mov eax, student\_ID[esi]; //Save the value of student\_ID[esi] in eax*

*cmp eax, 18000; //Judge whether the ID is less than 18000 or not*

*jl invalid\_ID; //if the ID is less than 18000, jump to "invalid\_ID"*

*cmp eax, 18999; //Judge whether the ID is greater than 18999 or not*

*jg invalid\_ID; //if the ID is greater than 18999, jump to "invalid\_ID"*

*jmp input\_student\_name; //If the input ID satisfy all the conditions, jump to "input\_student\_name"*

*invalid\_ID: //Function: to tell the user that the input student ID is invalid*

*lea eax, student\_ID\_warning;*

*push eax;*

*call printf; //Print "Warning: Invalid Student ID. Must be between 18000 to 18999." on the screen*

*add esp, 4;*

*jmp input\_student\_ID; //Jump to the "input\_student\_ID", let the user input the student ID again*

*//==============================name====================================*

*input\_student\_name: //Fuction: to ensure that all the input student names are valid*

*mov eax, current\_loop; //Save the value of current\_loop in eax*

*push eax;*

*lea eax, student\_name\_format;*

*push eax;*

*call printf; //Print "Enter Student Name [current\_loop]:" on the screen*

*add esp, 8;*

*lea eax, [edi];*

*push eax;*

*lea eax, string\_format;*

*push eax;*

*call scanf; //Input the student name which will be saved in student\_name[edi]*

*add esp, 8;*

*lea eax, [edi]; //Save the content of student\_name[edi] in eax*

*push eax;*

*call strlen; //Call the strlen fuction to return the length of the input student name to eax*

*add esp, 4;*

*cmp eax, 10; //Compare the length of the input student name with 10*

*jg invalid\_name; //If the length of the input student name is greater than 10, jump to the "invalid\_name"*

*add edi, 30; //Since the initialization of student name is student\_name[10][30], add 30 to edi every loop for next input*

*jmp input\_grade; //If the input name satisfy all the conditions, jump to "input\_grade"*

*invalid\_name: //Function: to tell the user that the input student name is invalid*

*lea eax, student\_name\_warning;*

*push eax;*

*call printf; //Print "Warning: Invalid Student Name. Must be 10 characters or less." on the screen*

*add esp, 4;*

*jmp input\_student\_name; //Jump to the "input\_student\_name", let the user input the student name again*

*//===============================grade==================================*

*add\_fail: //Function: to record the number of students who fail*

*inc fail; //Add 1 to variable fail*

*jmp add\_newline; //Jump to "add\_newline"*

*invalid\_grade: //Function: to tell the user that the input grade is invalid*

*lea eax, grade\_warning;*

*push eax;*

*call printf; //Print "Warning: Invalid Grade. Must be between 0 to 100." on the screen*

*add esp, 4;*

*jmp input\_grade; //Jump to the "input\_grade", let the user input the student grade again*

*input\_grade: //Fuction: to ensure that all the input grades are valid*

*mov eax, current\_loop; //Save the value of current\_loop in eax*

*push eax;*

*lea eax, Grade\_format;*

*push eax;*

*call printf; //Print "Enter Grade [current\_loop]:" on the screen*

*add esp, 8;*

*lea eax, grade[esi]; //Save the current address of grade in eax*

*push eax;*

*lea eax, int\_format;*

*push eax;*

*call scanf\_s; //input the grade*

*add esp, 8;*

*mov eax, grade[esi]; //Save the value of grade[esi] in eax*

*cmp eax, 0; //Compare the grade with 0*

*jl invalid\_grade; //If the grade is less than 0, jump to "invalid\_grade"*

*cmp eax, 100; //Compare the grade with 100*

*jg invalid\_grade; //If the grade is greater than 100, jump to "invalid\_grade"*

*cmp eax, 40; //Compare the grade with 40*

*jl add\_fail; //If the grade is less than 40, jump to "add\_fail"*

*jge add\_newline;*

*//===============================bubble sort===========================*

*sort: //Fuction: use the bubble sort to generate a sorted list from highest to lowest grade*

*mov ecx, loopCounter; //Save the value of loopCounter in ecx*

*dec ecx; //ecx minus 1*

*L1: //First loop*

*push ecx; //Protect the value in ecx*

*mov esi, 0; //Initialize esi equal to 0*

*lea edi, student\_name; //Save the head address of student\_name in edi*

*L2: //Second loop*

*mov eax, grade[esi]; //Save the value of grade[esi] in eax*

*cmp grade[esi + 4], eax; //Compare the value of grade[esi + 4] and grade[esi]*

*jle L3; //If the value of grade[esi + 4] is less than or equal to grade[esi], jump to "L3"*

*xchg eax, grade[esi + 4]; //Else, exchange the value of eax and grade[esi + 4]*

*mov grade[esi], eax; //Let the grade[esi] equals eax.*

*name\_swap: //Fuction: swap two students' name*

*mov eax, [edi]; //Save the content of student\_name[edi] in eax*

*xchg eax, [edi + 30]; //Exchange the value of eax and student\_ID[edi + 30]*

*mov [edi], eax; //Let the student\_name[edi] equals eax*

*mov eax, [edi+4]; //Save the content of student\_name[edi+4] in eax*

*xchg eax, [edi + 34]; //Exchange the value of eax and student\_ID[edi + 34]*

*mov [edi+4], eax; //Let the student\_name[edi+4] equals eax*

*mov eax, [edi + 8]; //Save the content of student\_name[edi+8] in eax*

*xchg eax, [edi + 38]; //Exchange the value of eax and student\_ID[edi + 38]*

*mov[edi + 8], eax; //Let the student\_name[edi+8] equals eax*

*mov eax, student\_ID[esi]; //Save the value of student\_ID[esi] in eax*

*xchg eax, student\_ID[esi + 4]; //Exchange the value of eax and student\_ID[esi + 4]*

*mov student\_ID[esi], eax; //Let the student\_ID[esi] equals eax*

*L3:*

*add esi, 4; //Add 4 to esi for each loop*

*add edi,30; //Add 30 to esi for each loop*

*loop L2; //The second loop ends*

*pop ecx; //ecx minus 1 for each loop*

*loop L1; //The first loop ends here*

*//============================reslult=================================*

*lea eax, result;*

*push eax;*

*call printf; //Print "List of Student IDs, Student Names and their Grades: " on the screen*

*add esp, 4;*

*mov ecx, loopCounter; //Save the value of loopCounter to ecx*

*mov current\_loop, 1; //Set the value of current\_loop to 1*

*mov ID\_address, -4; //Set the value of ID\_address to -4*

*lea edi, student\_name; //Move the head address of student\_name to edi*

*output\_result: //Fuction: Out put the sorted list as results*

*push ecx; //Protect the value in ecx*

*mov eax, current\_loop;*

*push eax;*

*lea eax, order;*

*push eax;*

*call printf; //Print "current\_loop." on the screen*

*add esp, 8;*

*add ID\_address, 4; //Add 4 to ID\_address for each loop*

*mov esi, ID\_address; //Save the value of ID\_address in esi*

*mov eax, student\_ID[esi]; //Save the value of student\_ID[esi] in eax*

*push eax;*

*lea eax, int\_tab;*

*push eax;*

*call printf; //Print "student\_ID[esi]\t" on the screen*

*add esp, 8;*

*lea eax, [edi]; //Save the content of student\_name in eax*

*push eax;*

*call printf; //Print "student\_name" on the screen*

*add esp, 4;*

*mov eax, grade[esi]; //Save the value of grade[esi] in eax*

*push eax;*

*lea eax, tab\_int;*

*push eax;*

*call printf; //Print "\t grade[esi]" on the screen*

*add esp, 8;*

*lea eax, newline;*

*push eax;*

*call printf; //Add a newline*

*add esp, 4;*

*inc current\_loop; //Add 1 to current\_loop for each loop*

*add edi, 30; //Add 30 to edi for each loop*

*pop ecx; //ecx minus 1 for each loop*

*loop output\_result; //"output\_result" Loop ends here*

*//==================================mean=============================*

*mov ecx, loopCounter; //Save the value of loopCounter to ecx*

*lea esi, grade; //Save the head address of grade in esi*

*addLoop: //Function: add all the grades together*

*mov eax, [esi]; //Save the value of [esi] in eax*

*add sum, eax; //Add the value of eax to sum*

*add esi, 4; //Add 4 to esi for each loop*

*loop addLoop; //"addLoop" ends here*

*mov eax, dword ptr[sum]; //Save the value of sum in eax*

*cdq;*

*div dword ptr[loopCounter]; //Return the result of eax/loopCounter in eax*

*mov average, eax; //Save the value of eax in "average"*

*mov eax, edx; //Save the value of edx (remainder) in eax*

*imul eax, 100; //Return the result of eax\*100 in eax*

*cdq;*

*div dword ptr[loopCounter]; //Return the result of eax/loopCounter in eax*

*mov fraction\_part, eax; //Save the value of eax in "fraction\_part"*

*mov eax, edx; //Save the value of edx (remainder) in eax*

*imul eax, 2; //Return the result of eax\*2 in eax*

*cmp eax, loopCounter; //Compare the value of eax and loopCounter*

*jge ceiling; //If eax is greater than loopCounter, jump to "ceiling"*

*cmp eax, loopCounter; //Compare the value of eax and loopCounter*

*jl floors; //If eax is less than loopCounter, jump to "floors"*

*ceiling:*

*inc fraction\_part //Add one to the fraction\_part*

*floors:*

*mov eax, average;*

*push eax;*

*lea eax, mean;*

*push eax;*

*call printf; //Print out integer part of the mean on the screen*

*add esp, 8;*

*mov eax, fraction\_part;*

*push eax;*

*lea eax, fraction1;*

*push eax;*

*call printf; //Print out fraction part of the mean on the screen*

*add esp, 8;*

*//========================Standard deviation=================*

*// Formula of Standard Deviation:*

*//s={[n(x1^2+x2^2+……+xn^2)-(x1+x2+……+xn)^2]/[n(n-1)]}^(1/2)*

*// In the code:*

*//s={(loopCounter\*square\_sum-sum\_square)/[loopCounter(loopCounter-1)]}^(1/2)*

*mov ecx, loopCounter; //Save the value of loopCounter to ecx*

*lea esi, grade; //Save the head address of grade in esi*

*add\_square:*

*mov eax, [esi]; //Save the value of [esi] in eax*

*mul eax; //Save the value of [esi]\*[esi] in eax*

*add square\_sum, eax; //Add the value of [esi]\*[esi] to square\_sum*

*add esi, 4;*

*loop add\_square; //Loop to get the sum of the grades square*

*mov eax, square\_sum; //Save the value of square\_sum (square\_sum=0 at first) in eax*

*mul loopCounter; //Save the value of square\_sum\*loopCounter in eax*

*mov square\_sum, eax; //Get the result of n(x1^2+x2^2+……+xn^2) and it is saved in "square\_sum"*

*mov eax, sum\_square; //Save the value of sum\_square (sum\_square=1 at first) in eax*

*mul sum; //Save the value of sum in eax*

*mul sum; //Save the value of sum\*sum in eax*

*mov sum\_square, eax; //Get the result of (x1+x2+……+xn)^2 and it is saved in "sum\_square"*

*mov eax, dividend; //Save the value of dividend (dividend=0 at first) in eax*

*add eax, square\_sum; //Add square\_sum to eax*

*sub eax, sum\_square; //eax minus sum\_square*

*mov dividend, eax; //Get the result of [n(x1^2+x2^2+……+xn^2)-(x1+x2+……+xn)^2] and it is saved in "dividend"*

*fld divisor; //Give a real number (divisor) to float stack, divisor=1 at first*

*fimul loopCounter; //Get the result of divisor\*loopCounter*

*fimul loopCounter; //Get the result of divisor\*loopCounter^2*

*fst divisor; //Save the result of loopCounter^2 back to divisor*

*fild loopCounter; //Give an integer number (loopCounter) to float stack*

*fsubr divisor; //Get the result of loopCounter^2-loopCounter*

*fst divisor; //Get the result of [n(n-1)] and it is saved in "divisor"*

*fld divisor; //Give a real number (divisor) to float stack, divisor=loopCounter^2-loopCounter now*

*fidivr dividend; //Get the result of dividend/divisor*

*fst deviation; //dividend/divisor is [n(x1^2+x2^2+……+xn^2)-(x1+x2+……+xn)^2]/[n(n-1)] and it is saved in deviation*

*fld deviation; //Give a real number (deviation) to float stack, deviation=s^2 now*

*fsqrt; //To get the final answer s, we need to extract the square root of deviation*

*fst deviation; //Save the Standard Deviation in "deviation"*

*sub esp, 8; //Reserve stack for a double in stack*

*fld dword ptr deviation; //Load deviation*

*fstp qword ptr[esp]; //Convert to double and store*

*lea eax, deviation\_format;*

*push eax;*

*call printf; //Print "Standard deviation is %.4f" on the screen*

*add esp, 12;*

*//==============================fail=========================*

*mov eax, fail;*

*push eax;*

*lea eax, fail\_format;*

*push eax;*

*call printf; //Print out how many students failed*

*add esp, 8;*

*//============================end============================*

*finish:*

*lea eax, end;*

*push eax;*

*call printf; //Print "Program ends." on the screen*

*add esp, 4;*

*}*

*system("Pause");*

*return 0;*

*}*