**CMPE 50 – Fall 2020, Tarng**

**Lab #2 – Function overloading, Arrays & multidimensional Arrays.**

**Instructions**: Before you leave the lab, you should submit your answers through Canvas. Please submit your answers (.cpp files with output embedded as comments and with appropriate documentation/comments) even if you couldn’t complete/run them. People who want to leave before the end of the lab need to get approval by showing the work and result to the on-duty ISA or instructor.

*For each of the exercises, you need to write an algorithm and pseudo-code, in a comment section of the source code (.cpp). The algorithm and pseudo-code can cover only the key portion of the solution. There is no need to write algorithms for trivial code. Some examples of algorithms are given at the end of this assignment.*

**Exercise 1 – Function Overloading**

Write an overloaded function *Add( )* that takes two parameters of the same type. The possible types of the arguments are type double, int, and char array[ ]. *(Hint: define three Add().)*

Write a test driver, i.e., the main() function, and ask the user if he/she would like to add two doubles, two int, or two character arrays. Adding two strings is to concatenate them. In this case just output the two arrays one after the other using two cout statements with no space in between.

**Exercise 2 (Ch. 7 - Arrays)**

Write a program that will read up to 10 letters (characters) into an array and write the letters back to the screen in the reverse order. The program should prompt the user to input all values and can determine whether the input has ended by a punctuation mark such as ‘.’ (make sure that you’ll include meaningful and intuitive messages to the user). Please note that C++ stores a string with an extra NULL character at the end to terminate the string so you can’t just output the array from the last index back to 0.

For example, if the input is *abcde*, the output should be *edcba*. Include appropriate messaging (UI, i.e., user interface) to the user.

**Exercise 3 (Multidimensional Arrays)**

Write a program that asks a student for his/her grades assessing his/her C++ programming skills. The student gets a separate grade for each homework assignment and the lab exam associated with it. In order for the student to pass the class, he/she needs to achieve 50% or better in all possible “dimensions”, i.e.,

* >= 50% average in homework assignments
* >= 50% average in lab exams

Assume that the student has been given 5 “sets” of homework assignments and lab exams, respectively, and that the scores are given in percentages.

The program should get input from the screen and do the following:

1. Calculate the best, worst, and average grades for each of the two components (hw assignments and lab exams).
2. Calculate the overall grade of the student with every component carries the same weight.
3. Decide whether the student passes the class or not. If the student cannot pass the class, the program should output the reason why.
4. Output on the screen the input and results so that they look similar to this (use appropriate tools for formatting I/O):

1 2 3 4 5 BEST WORST AVG

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

HW 100.00 90.00 80.00 100.00 100.00 100.00 80.00 94.00

LAB 100.00 100.00 70.00 60.00 90.00 100.00 90.00 84.00

AVG 100.00 95.00 75.00 80.00 95.00 Overall: 89

(option 1) Congratulations! You passed the class!

(option 2) I’m sorry to inform you that you failed the class, because your performance in <enter\_component\_here> was <50%. Try again next semester.

1. (Optional) Output the same information as above to a file. The user should input the file name during run time). Do this part only if you know how to write to a file. Will teach in the class later.

**Note:**

You can use the following cout formatting statements:

cout.setf(ios::fixed); ; for fixed point notation

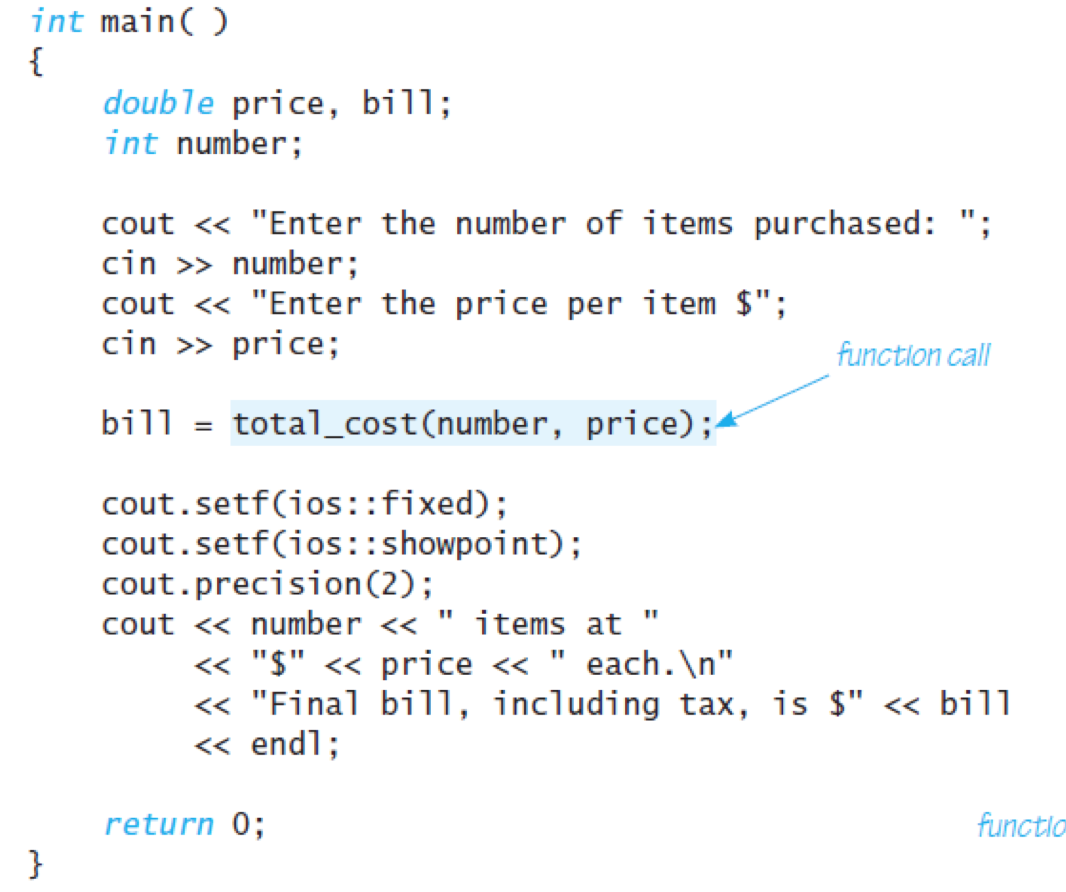
cout.setf(ios::showpoint); ; for showing the decimal points

cout.precision(2); ; for showing two decimal places

cout.width(4); ; set the output width to be 4. Need to be

; called for every single cout statement

See Display 4.3 for the usage of these formatting statements, as follows.



**Recommended (optional, no submission needed):**

* *All* self-test exercises of Ch. 5, 7
* Programming Projects: 5.1, 5.2, 5.4, 5.8- 5.10, 5.12, 5.16, 7.1, 7.4, 7.6

## Algorithm for Exercise 2

1. Declare a char array. *Question: what should be the array size so it can store up to 10 letters?*
2. Declare an int variable *size* and initialize it to 0. Enter a loop to read a char from the console at a time. Check if the char is ‘.’, if so exit the loop and go to step 3. If not, copy the char to the array element and increase the variable *size*. Check the size to ensure the array does not exceed the boundary. If so, output an error message and exit the program.
3. Iterate the index from (*size*-1) down to 0 and print the array element to the console.

## Pseudo-code for Exercise 2

1. const int ARRAY\_SIZE = 100; // define upper limit of the array size
2. char ch, array[ARRAY\_SIZE];
3. int size = 0;
4. cin >> ch;
5. while (ch != ‘.’)

{

array[size] = ch;

size ++;

if (size >= ARRAY\_SIZE)

{

Array limit is reached.

exit(1); // exit the program with status 1

}

cin >> ch;

}

1. for (int i = size-1; i >= 0; i--)

{

cout << array[i];

}

## Algorithm for Exercise 3

1. Declare a two dimensional double array *val[2][5]*. *Val[0][]* is used to store HW assignments and *val[1][]* is used to store exams.
   1. Alternatively, since we need to store the best, worst, and average scores on each component, we can extend the width of the array as *val[[2][8]* so we can use *val[0][5]*, *val[0][6]*, *val[0][7]* to store the best, worst, and average HW, respectively, and so on.
2. Loop for five times and input the HW assignment to the *val[0][]*.
3. Loop for five times and input the exams to the *val[1][]*.
4. Declare three double variables for HW best, worst, and average; and three double variables for Exam best, worst, and average. If you declare *val[2][8]* as in 1.a, skip this step.
5. Find the best, worst, and averages of HW and Exams.
   1. Find the best: iterate the five scores by comparing each with the variable best. If greater than the best, update best to be the current HW assignment.
   2. Find the worst: similar but change “greater” to be “smaller” as in step 5.a
   3. Find the average: add the five scores and divide the sum by five.
6. Declare a double variable for overall grade. Calculate the overall grade by averaging the two averages of the two sets of the scores.
7. Declare a bool variable *is\_passed*. If both averages (HWs and Exams) are equal or above 50, then *is\_passed* is set to true. Otherwise, it is false.
8. Output the scores in the format specified.

## Pseudo-code for Exercise 3 (Paritial)

1. const int ASSIGNMENT\_NUM = 2, SET\_NUM = 5;
2. double val[ASSIGNMENT\_NUM][SET\_NUM];
3. Input the HW assignments:

cout << “Please enter five HW scores [0 – 100]: “;

for (int i = 0; i < SET\_NUM; i++)

{

cin >> val[0][i];

}

1. Find the best HW scores:

double hw\_best = 0;

for (int i = 0; i < SET\_NUM; i++)

{

// Find the intermediate best HW score

if (val[0][i] > hw\_best)

{

hw\_best = val[0][i];

}

Find the intermediate worst HW score …

Find the intermediate average HW score …

}

1. Continue …