The University of Alabama in Huntsville
Electrical and Computer Engineering
Project 3 (10 points)

Submit Your Solution Using ANGEL by Noon, Monday January 28, 2013

<Project 3 Description>

There is one file that needs to be downloaded for this program. Download the file Project_03_base.cpp from ANGEL into your CPE112_SPR13/Project_03 directory.

For this project, modifications (starting on page 3) will be made to Project_03_base.cpp. Therefore, make a copy of Project_03_base.cpp (**cp Project_03_base.cpp Project_03.cpp**) and make the changes to the file **Project_03.cpp**. Note: This will overwrite the Project_03.cpp file that you placed in your Project_03 directory at the end of Project 2.

After making one or two modifications, compile the modified code to verify that there are no syntax errors. If there are syntax errors, correct them and compile again. Once the program compiles, make one or two more modifications and compile the program again.

After completing modifications #0 through #11 and the program successfully compiles, run it and verify that the output for your modified program matches the output from the sample solution executable.

Once you are satisfied with your programs execution, submit your Project_03.cpp file using ANGEL.

To compile and run your program, perform the following three steps

1. To compile type in the command **`g++ Project_03.cpp –o Project_03`**
2. To run the program type **./Project_03** at a command prompt in the terminal
3. Enter in values as prompted.

< Sample data to use for program verification >

- All data provided below appears in the order in which it should be entered.
- Commas are used in this document to separate the values entered. The commas are not entered.

Test Case 1: `10, 10, 5, 10, 10, 20, 5, 5, 3.25, -1`
Test Case 2: `5, 10, 50, 1, 2, 9.5, -1`

A sample solution executable and a program comparison script are available for this project. For information on how to use these files, look at the Project_03_slides available from ANGEL.

***NOTE #1:* Read the modification step instructions carefully and perform the task required. Most of the modifications involve completely removing the comment line indicating the modification and replacing that line with some other code.**

***NOTE #2:* Make sure that you do not change the order in which the information is entered. An automatic script is used to process all lab submissions, and if the order of the input information is modified, the script will not work properly with your program.**

***NOTE #3:* There are two distinct parts of code in this project, and comment statements are used to separate the two parts.**

<Project 3 C++ Concepts Explained>

This program uses some C++ concepts that need to be further explained. The introduction of the concepts and the subsequent explanation is meant to be an overview of the concepts. These concepts will be explained in greater detail once the material is reached in the text.

The following C++ concepts are introduced in this project:

global constants
cout and cin statements
void function (Chapter 8, and some brief information earlier), and
if statements (Chapter 5)
while statements (Chapter 6)
iomanip header file
setprecision(), and
setw()
Summation variable

Global Constants: Global constants are declared in the area above the start of the main program [int main()]. These constants maintain their value throughout the entire running of the program, and their value cannot be changed. Declaring constants allows for easier understanding of the code and for easier modifications if any are required in the future. Constants declared as global are available to any part of the program (main and any functions) below where they are declared.

cout and cin statements: These statements are the standard output (the terminal) and standard input (keyboard) devices. cout is used to output information to the standard output device, and cin is used to obtain information from the standard input device.

Void Function: There are two types of functions – void and value returning. Void functions do not return a function value. However, they can return values back to the part of the program that invoked (called) the function. This transfer of information from the function back to the caller takes place using reference parameters. Functions are used to add clarity to a program. If a segment of code is to be used in several different places in a program, then that segment of code should be written as a function. Then a function call is used to invoke the function and execute the segment of code.

if Statements: If statements provide a way to branch the execution of the code in a program. There are two possible branches available with if statements. One branch of code is executed when the test expression is true, and the other branch of code is executed when the test expression is false.

while Statements: While statements provide a way to repeat the execution of a block of code inside the while statement so long as the loop control expression evaluates to true when evaluated prior to an iteration.

The iomanip header file: This header file contains definitions for the output manipulators setprecision() and setw(). If you try to use setprecision() and setw() without the iomanip header file, you will see syntax errors.

setprecision(): Setprecision sets up how many decimal places are to be output by the computer. In this program we are dealing with dollar amounts, so that is the reason for using setprecision(2). This output manipulator is set for all outputs once it has been used.

setw(): Setw sets the field width for the next item to be output. When the computer prints the information, it is right justified in the field width specified. In this program the field width is specified as 20 characters. Setw is valid only for the next item output.

Summation variable: A Summation variable may be used in conjunction with a looping statement, such as the **while** statement, to accumulate a total value from individual calculations performed on each iteration or pass through the loop. Note that in the modified program, Summation variable is initialized to an appropriate value prior to the loop.

<Project 3 Modifications>

The following are the modifications to be made to the program. Using any text editor, open the source code file that is to be modified. The editor **gedit** is recommended for use on eagle. This editor is invoked at the prompt by typing the command: **gedit Project_03.cpp &** where Project_03.cpp represents the name of the file created by copying the downloaded Project_03_base.cpp file. **The & runs the editor in the background of the terminal window so that the terminal window stays active for other commands.**

Modification #0: This modification places your name, your lab section and project due date in the header part of the program. Replace the phrase **Modification #0** with the appropriate information.

Modification #1: This modification adds the iomanip header file to the program. **Replace the comment statement for modification #1** with the following line (be sure to remove the // characters that start the comment line for modification #1):

```
#include <iomanip>
```

Modification #2: This modification adds two more variable declarations to the program. **Replace the comment statement for modification #2** with the following two lines:

```
float over_wages;    // Amount of the wages that is overtime
float reg_wages;     // Amount of wages that is regular pay
```

Modification #3: Note: the next modification in the file is noted as modification #5. Modification #3 is farther down in the file. This modification adds the following if-then-else-if statement to the program. This if statement allows for determining how much of the wages earned comes from regular pay and how much from overtime. **Replace the comment statement for modification #3** with the following code:

```
    if (hours < MAX_HOURS)
    {
         reg_wages = wages;
         over_wages = 0.0;
    }
    else
    {
         // calculate the amount of the wages that comes from
         // regular pay, and the amount that comes from overtime.
         reg_wages = payRate*40.0;
         over_wages = wages - reg_wages;
    }    // end of if then else statement
```

Modification #4: This modification changes the cout statements used to output the information. **Replace the two comment statements for modification #4** with the following lines. **Also, remove (as indicated) the 4 cout** statements that appear just below the modification #4 comment lines.

```
cout << setprecision(2) << fixed;
// break up long cout statement into three lines.  Break at insertion op.
cout << setw(20) << "Employee: " << empNum << endl
     << setw(20) << "Pay rate: " << payRate << endl
     << setw(20) << "Hours: " << hours << endl;
cout << setw(21) << "Regular wages: $" << reg_wages << endl;
cout << setw(21) << "Overtime wages: $" << over_wages << endl;
cout << setw(21) << "Total wages: $" << wages << endl;
```

The following modifications are for the second part of the program

Modification #5: This modification adds a Summation variable to the program. This modification is in the variable declaration section for Part II of the program. **It is up near the start of main**. Replace the comment statement for modification #5 with the following line:

```
float totalPrice;          // Total price of all ordered parts
```

Modification #6: This modification sets the precision of decimal numbers to two places. **Replace the comment statement for modification #6 and the line below it** with the following line:

```
cout << fixed << showpoint << setprecision(2);
```

Modification #7: This modification initializes the Summation variable. **Replace the comment statement for modification #7** with the following line:

```
totalPrice = 0.0;          // Initialize summing variable to zero
```

Modification #8: This modification creates a priming read prior to the while loop. This will allow the loop to detect a negative parts number, the quit signal to stop looping and print the total price. **Replace the comment statement for modification #8 with the following line of code, and delete the cout line below the comment.**

```
cout << "Enter the part number (or a negative number to quit):";
```

NOTE: Modifications #9 and #10 must both be made before compiling the program. Making modification #9 only or modification #10 only will result in a compile error.

Modification #9: This modification adds the while loop, loop control expression, and the start of block symbol {. Be sure not to forget adding the left curly brace!! **Replace the comment statement for modification #9** with the following code:

```
while (partNumber >= 0)       // While part number is valid
{                             // Start of the while loop
```

Modification #10: This modification will update the summation value at the end of each pass through the while loop, prompts the user for the next part number and end the while loop. **Replace the comment statement for modification #10** with the following code:

```
   totalPrice = totalPrice + partsPrice;

   cout << endl << "Enter the part number (or a negative number to quit):";
   cin >> partNumber; // read in the next part number
   cout << partNumber << endl;
} // End while(partNumber >=0)
```

Modification #11: This modification prints out the final total for all parts ordered. **Replace the comment statement for modification #11** with the following code:

```
cout << endl << "Total price of all parts ordered is: $";
cout << totalPrice << endl << endl;
```

After making the modifications, compile your program. Once it compiles, run it and verify that results obtained from your modified program match those of the sample solution provided (see the first page). **Once you are satisfied with the solution, submit your source code to the drop box for project 3.**