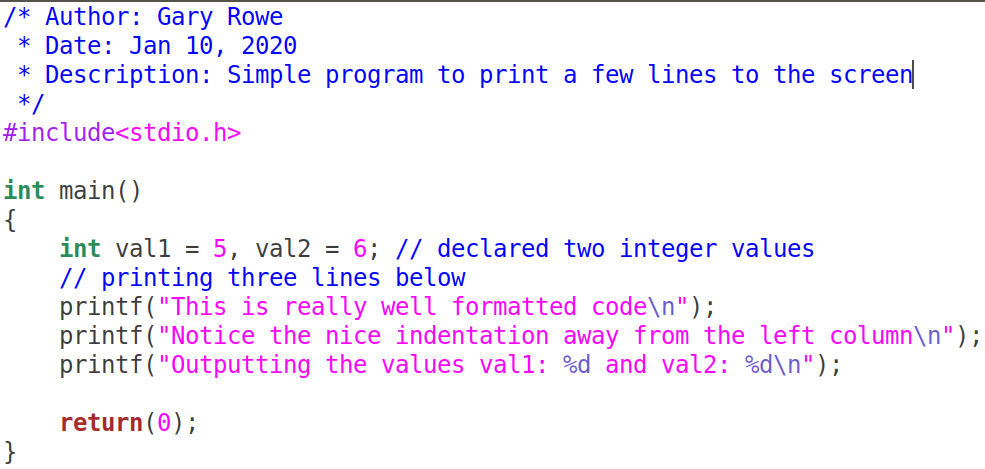
**Student Name:** **Weight: 85**

**Student ID:** **Marks: \_\_\_\_\_\_\_\_\_\_**

**CREATE AN EMPTY DOCUMENT TO SUBMIT YOUR SOLUTIONS. DO NOT USE THIS DOCUMENT TO SUBMIT YOUR ANSWERS. YOU WILL LOSE 10% FOR DOING SO!!!**

Assignment 1:

Your C files should be properly ***formatted*** with ***indentations*** that enhance code ***readability***. Example of properly formatted code:

**Important:**

**CREATE AN EMPTY DOCUMENT TO SUBMIT YOUR SOLUTIONS. DO NOT USE THIS DOCUMENT TO SUBMIT YOUR ANSWERS. YOU WILL LOSE 10% FOR DOING SO!!!**

* On your Ubuntu VM guest machine create a folder called ITSC202.
* Inside the ITSC202 folder create a subfolder called **A1**.

A1 is the folder you will use for all the C files for **Assignment 1**.

* **All Assignments must have an image showing the execution of the program.**

Problem 1 (15):

You have been asked by a math teacher to write a program for her students to calculate the sum and average of 10 numbers. The numbers can be either floating points or integer numbers.

The criteria are as follows:

* Create a file called **m1p1.c**
* The program will ask for 10 numbers.
* Your program will print the **sum**, **average** of those values entered.
* Examples of output are shown below:

**You will submit the C source code and 1 sample run of your program**

[/home/**noobhack1**/a1/]$./**m1p1**

Enter a number: 30

Enter a number: 40

Enter a number: 5

Enter a number: 0

:

:

Average is: 25.0000

Sum is: 75.0000

[/home/**noobhack1**/assignment1/]$./**m1p1**

Enter a number: 50

Enter a number: 16

Enter a number: 22

Enter a number: 2

Enter a number: 0

:

:

Average is: 22.5000

Sum is: 90

**Notice that the Average values are always floating pointer numbers.**

Problem 2 (15):

You have been asked to write a program that prints a message based on a number entered by the user. Each number will be associated to a name. Your menu can include different names or topics of interest to you.

The criteria are as follows:

* In the folder assignment 1, create a file called **m1p2.c**
* The program should print 5 – 10 names on the screen with a unique number associated to that name. The numbers do not need to be sequential but they must be unique.
* The user will be asked to enter a number from 1 to as many entries exist in your printed menu of names
* The program will echo, the number the user entered.
* Examples of output is shown below:

**You will submit the C source code and 1 sample run of your code.**

[/home/student/ITSC202/m1]$./**m1p2**

1. Reconnaissance
2. Weaponization
3. Delivery
4. Installation
5. Exploitation
6. Privilege Escalation
7. Command and Control
8. Quit

Enter the number associated with your choice: 5

You entered the number: 5

Thanks for accessing the program … Quitting!!!

[/home/student/ITSC202/m1]$./**m1p2**

1. Reconnaissance
2. Weaponization
3. Delivery
4. Exploitation
5. Installation
6. Command and Control
7. Action on Objective
8. Quit

Enter the number associated with your choice: 3

You entered the number: 3

Thanks for accessing the program … Quitting!!!

**NOTE: We have not had a lesson on how to match user input to a word. If you know how to do so, you can modify the program so that it prints the word associated with the user input.**

Problem 3 (15):

Write a program that prints the pattern shown below:

The criteria are as follows:

* In the folder Assignment 1, create a file called m1p3.c.
* Use only 2 printf statements to complete this question.
* For a challenge complete the program using only 1 printf statement

**You will submit ONLY the C source code and an image showing the output**

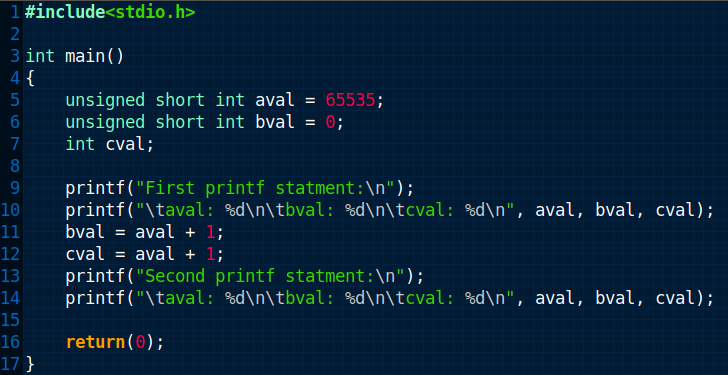
[/home/student/Assignment1/]$./**m1p3**



Problem 4 (15):

Copy the following C code, compile and execute it; then do the following:

1. Analyze the program to ensure you understand what is happening
2. Comment the code with meaningful statement where needed.
3. **Submit your commented code and answers to the questions below**
4. Answer the following questions:
   1. At **lines 5, 6 and 7**, what are the values of each variable (**aval, bval and cval**)?
   2. At line 7, how did cval get its value?
   3. What does the qualifier **unsigned** do?
   4. What does the qualifier **short** do?
   5. At lines 11 and 12, 1 is added to aval. **Why does aval remain unchanged?**
   6. After the operations on lines 11 and 12, why are the values of bval and cval different?
      1. **HINT: Think about how much data each data type can hold.**



**Investigate other keywords like long, const and static. Ask yourself, how can I use these keywords.**

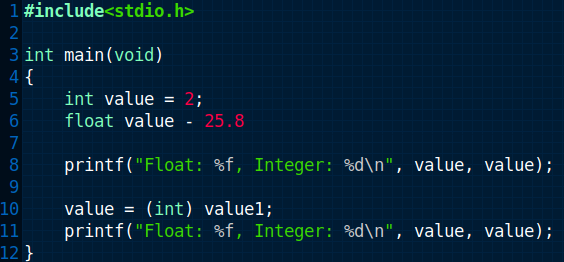
Problem 5 (25):

Analyze the following lines of code. Fix the code so that it will run. You will likely need to

* Rename variables
* Determine if syntax is correct
* Ask Google how casting in C works

In **point format** explain the steps you had to take to fix the code. Example:

1. Changed variable name to match …
2. Terminated statements at line …



Answer the following questions, **after fixing the code**:

1. What operation is line 10 attempting to accomplish?
2. Why is the output of lines 8 and 11 different?