Intro to C

1. Below, you will find two versions of a program that does the following when it runs:
- declares and initializes a variable to represent the length of the side of a square (assumed to be in metres)
- prints the area of the square (in square metres) on the screen

List at least 5 reasons why the second version of the program could be considered better than the first (and would therefore receive more marks). Note: both programs compile and run correctly!

Version 1:

```
#include <stdio.h>
int main( void ) {
double l;
double a;
l = 3.2;
a = l * l;
printf( "Answer: %f\n", a );
return 0;
}
```

Version 2:

```
/*
 * Author: Joe Gear
 * Purpose: define and initializes a variable representing the
 *          the  of side of square (in metres) and
 *          prints area of square on screen (in square metres).
 */

#include <stdio.h>

int main( void ) {
    double length;
    double area;

    length = 3.2;

    area = length * length;
    printf( "Area of square: %f square metres\n", area );

    return 0;
}
```

- Version 1 is missing:

- **- documentation**
- **- descriptive variable names**
- **- indentation of code within the curly brackets**
- **- white space to improve readability**
- **- descriptive output including units**

2. Suppose that in lab, students were asked to write a program that: declares a variable for the area of a plot of land in acres and initialize it to 567.2 and prints the area of the same plot of land in square metres.
Assume that 1 acre is equivalent to 4046.85642 square metres. One student, Pat, submitted the following code:

```
/*
 * Author: Pat Gear
 * Purpose: ...
 */

#include <stdio.h>

int main( void ) {
   double area1 = 567.2;
   double area2;

   area2 = area1 / 4046.85642;
   printf( "Area in square metres: %f\n", area2 );

   return 0;
}
```

a) The program does not produce the right output. Explain what you would do to fix it.

Calculation should be: `area2 = area1 * 4046.85642;`

b) Assuming that the lab is marked out of 10, how many marks would you award Pat for this work? Give Pat some feedback for getting a better mark in the next lab.

- add documentation (purpose)
- use descriptive variable names ie. area_in_acres, area_in_meters (area1 and area2 are easily mixed up)
- good programming practice to define `4046.85642` as a symbolic constant as it does not change

3. Recall that constants, such as the acceleration due to gravity, are best represented using a symbolic constant declared with the `#define` directive. For example: `#define GRAV_ACCEL 9.81`

Write a program that declares and initializes a variable for the price of an article (before taxes). You are free to choose the price of the article. The program should calculate and print the price of the article including provincial sales tax (PST) at 7% and goods and services tax (GST) at 5%. Use *Version 2* in Q1 above as a model. Here's a start:

```
/*
 * Author: CSC 111 Instructor
 * Date: Jan 1, 2020
 * Purpose: Defines the price of an article before tax
 * and prints the cost of the article including GST and PST.
 */

#include <stdio.h>

#define PST 0.07
#define GST 0.05

int main( void ){
  double price_before_tax = 15.67;
  double price_after_tax;

  price_after_tax = price_before_tax * ( 1.0 + PST + GST );
  printf("Price after tax: $%.2f\n", price_after_tax);

  return 0;
}
```

4. In this exercise we will investigate the use of format specifiers to control the way that output appears on the screen. When printing values of type double, format specifiers have the following form, in general:

```
"%-+X.Yf"
```

As you read through the following explanation of how to use format specifiers, label the pieces of the image above to help you remember what they control.

`Y` represents a positive integer that controls the number of decimal places that are displayed.
If `Y` is not specified, the default number of decimal places is 6.

`X` represents a positive integer that controls the width of the field in which the number is to be displayed.
By "width of the field", we mean the number of consecutive spaces allocated to display the output on the screen.
If `X` is not specified or too small, the number is displayed in a field just wide enough to print the number including the decimal portion.

The number printed is aligned on the right hand side of the field, by default.
If there's a "`-`" sign in front of the `X`, the number is aligned on the left hand side of the field.

If there's a "+" sign in front of the `X`, the sign of the number is always printed. So, if the number is positive, a `+` sign will be printed immediately preceding the number.

Suppose the following variable declarations have been made:

```
double a = 4.5236643231;
double b = -6.43864176;
```

Determine the output from each of the following `printf` statements. The first two are done for you – make sure you understand them before you move on.

`printf` statement	**output**											
`printf( "%f\n", a );`	4	.	5	2	3	6	6	4				
`printf( "%f\n", b );`	-	6	.	4	3	8	6	4	2			
`printf( "%.3f\n", a );`	4	.	5	2	4							
`printf( "%.5f\n", b );`	-	6	.	4	3	8	6	4				
`printf( "%6.2f\n", a );`			4	.	5	2						
`printf( "%7.1f\n", b );`				-	6	.	4					
`printf( "%-6.1f%4.1f\n", a, b );`	4	.	5				-	6	.	4		
`printf( "%-7.1f%+5.1f\n", b, a );`	-	6	.	4					+	4	.	5

5. Write a program that declares and initializes 3 variables for the length, width and height of a solid steel rectangular bar (in metres). You are free to choose your own values for length, width and height. The program must compute the mass of the bar in kilograms assuming that the density of the steel is 7850 kg per cubic metre and print it to the screen to 3 decimal places.
Recall that: mass = density * volume. Try to look as little as possible at the solutions to the previous problems while doing this!

```c
/*
 * Author: CSC 111 Instructor
 * Date: Sep 14, 2019
 * Purpose: defines and initializes a length, width & height of steel
 * rectangular bar (in metres) and computes mass of bar in
 * kilograms assuming density of STEEL_DENSITY kg/cu.m.
 */

#include <stdio.h>

#define STEEL_DENSITY 7850.0

int main( void ){
 double length = 67.3;
 double width  = 3.1;
 double height = 2.7;
 double mass;

 mass = ( length * width * height ) * STEEL_DENSITY;

 printf( "Mass of bar = %.3f kg\n", mass );

 return 0;
}
```