Formatted Printing

1. **Writing a Program to format the displayed values on the screen.**
   a) Open an editor (like Notepad)
   b) Type the following C program: [Note that copying and pasting this code to the editor may result to errors, due to the formatting in this document. Thus, please re-type]

```
#include <stdio.h>
int main()
{
       printf("Integers:\n");
       printf("1=>%5d\n", 4);
       printf("2=>%2d\n", 34);
       printf("3=>%2d\n", 3456);
       printf("4=>%2.5d\n",123);
       printf("5=>%8.5d\n",123);
       printf("6=>%+d\n",123);
       printf("7=>%dMARK\n",123);
       printf("8=>%-8.5dMARK\n",123);

       printf("\n\n");
       printf("Floating Point Values:\n");
       printf("9=>%f\n", 0);
       printf("10=>%f\n", 1.55);
       printf("11=>%3.3f\n",2.5);
       printf("12=>%3.3f\n",234.34567);
       printf("13=>%2.8f\n",234.34);
       printf("14=>%-3.5fMARK\n", 3124.45);
       printf("15=>%+10.5fMARK\n", 24.45);
       printf("16=>%+010.5fMARK\n", 24.45);
       printf("17=>%.3f\n", 1 / 3);
       printf("18=>%.3f\n", 1 / 3.0);
       printf("19=>%.3f\n", 2.0 / 3);

       printf("\n\n");
       printf("Strings:\n");
       printf("20=>%6.3s\n", "what a long string!");
       printf("21=>%10sMARK\n", "what a long string!");
       printf("22=>%-50sMARK\n", "what a long string!");

       return 0;
}
```

   c) Save your program using your chosen filename. Make sure that your extension is .c
   d) To compile your program, from the console (or command prompt) type
   gcc -pedantic <filename.c> -o <outputfilename>

   where: <filename.c> is your C program
   <outputfilename> is the name of the resulting executable file
   Example: gcc -pedantic sampleprg1.c -o sampleprg1.exe

   Note : -pedantic is an option that specifies that your program will be compiled strictly. This option will be used during your project demo too.

   e) If your source code has no errors, you will see in this directory your executable file, otherwise error messages will be shown. If there are errors, check the code again; you might have mis-typed something. Debug and compile your program again.

f) Run your program in the console (command prompt). Do not double-click the executable file. Instead, type your executable filename in the prompt.
Example: sampleprg1.exe

Name ____________________ Section ________

g) **Print this page** and write your answers to the questions below based on your observation and understanding of the effect of your program. [This page, with your answers, will be submitted.]

1. What does the 5 in **%5d** mean?

2. What does the 0 in **%05d** mean?

3. What does the + in **%+5d** mean?

4. What does the - in **%-5d** mean?

5. What is the use of **\n**?

6. What does the 6 in **%6.4f** mean?

7. What does the 4 in **%6.4f** mean?

8. Why is the result in number 17 and number 18 not the same?

9. When C displays a floating number to follow a certain number of decimal place (precision), does it use the rounding rules in Math? If no, explain what it does.

10. What does the 6 in **%6.3s** mean?

11. What does the 3 in **%6.3s** mean?

12. What does the - in **%-6.3s** mean?

2. **Code Readability.** The purpose of this exercise is to show that the compiler sees only a stream of characters. Although the compiler is not interested in neatness, for the human reader, proper formatting of source code is crucial. **Sloppy code is hard to read and maintain.** The following program although not very readable, should compile and execute.

   a) Copy the program **as is**. Compile using gcc and execute in the command prompt.

```
#include <stdio.h>
int main(
) {int I=1;
float  qx,
         zz,
    tt;
printf("***%d%d%d%d%d%d%d***\n", I,I,I,I,I,I,I); printf ("gimme 3"
);scanf("%f%f%f", &qx, &zz
,&tt)
;printf("a is=%f\n", (qx+tt+zz)/3.0);
printf("***%c%c%c%c%c%c%c***\n", I,I,I,I,I,I,I); return 0;}
```

   b) Now, type the following code following the same layout/format as found below. Save as another file (different filename from above). Compile using gcc and execute in the command prompt.

```
#include <stdio.h>

int main()
{
    int    i = 1;
    float  fQ1,
           fQ2,
           fQ3;

    /* Displaying intro design */
    printf("***%d%d%d%d%d%d%d***\n", I,I,I,I,I,I,I);

    printf ("Enter 3 numbers: ");
    scanf ("%f%f%f", &fQ1, &fQ2, &fQ3);
    printf("Average is=%f", (qx+tt+zz)/3.0);

    /* Displaying closing design */
    printf("***%c%c%c%c%c%c%c***\n", I,I,I,I,I,I,I);

    return 0;
}
```

Name ______________________ Section __________

c.) **Print this page** and write your answers to the questions below based on your observation and understanding of the effect of your program. [This page, with your answers, will be submitted.]

1. Does the second version give the same result as the first version, given the same set of input values?

2. Did changing the displayed message/s allow you (as the user) react / interact better to the program? Or the message did not matter?

3. Did changing the form and names in the variables allow you (as the programmer) understand better the purpose of these identifiers?

4. Did the indentions, spacing, and alignment help you (as the programmer) understand the code better?

5. Do you think that lessening the lines of the code, by just putting multiple statements in the same line, can make the source code more efficient?

6. Would lessening the number of lines of the code, by just putting multiple statements in the same line, make the source code readable?

7. What did you realize with the purpose of the conversion characters (%d and %c) in the printf statement?

8. From what you observed in the two programs, what do you promise to do when you write programs so that they are readable?