ME-172
Computer Programming Language Sessional
Assignment No. 1

Abu Sayeed Roni, ID: 1710065
Dept: ME, Section: B-2

February 24, 2020

Problem 1

Write a progam to find the area of a circle. [NOTE: Radius should be scanned from the keyboard.]

Solution:

Formula used:

$$A = \pi r^2$$

Value of π used in the following code is rounded to four decimal places. For our purposes this should do just fine.

```
/**
 * Calculates the area of a circle
 */

#include <stdio.h>
#include <math.h>


int main(void)
{
    const float pi = 3.1416;
    float radius, area;

    // Take user input for radius.
    printf("Enter radius: ");
    scanf("%f", &radius);

    // Calculate area.
    area = pi * pow(radius, 2);

    // Print output.
    printf("Area: %.2f\n", area);
}
```

Saving the file with name `circle.c`. Then Compiling the source file with the following command:

```
$ clang -o circle circle.c -lm
```

An executable binary file with name `circle` will be created if everything goes right.

Output

Running the program with the following command:

```
$ ./circle
Enter radius: 3.5
Area: 38.48
```

Problem 2

Write a progam to compute average of four user given numbers. [NOTE: Numbers can be of integer or floating types.]

Solution:

Formula used:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

n, in our case, just happens to be 4, though it could've been anything (any resonable positive integer) and we could still make the following code work just by setting `count` to whatever n would've been. We have this flexibility because of the better design decision taken instead of just *hard-coding* four separate variables.

```
/**
 * Computes average of four user given numbers.
 */

#include <stdio.h>


int main(void)
```

```
{
    // Allocating memory to store 4 numbers.
    const int count = 4;
    float numbers[count];

    // Prompt user for input
    printf("Enter four numbers: \n");
    for (int i = 0; i < count; i++)
    {
        scanf("%f", numbers + i);
    }

    // Calculate average
    float sum = 0;
    for (int i = 0; i < count; i++)
    {
        sum += numbers[i];
    }
    float avg = sum / count;

    // Print output
    printf("Average of ");
    for (int i = 0; i < count; i++)
    {
        if (i == count -1)
        {
            printf(", and ");
        }
        else if (i)
        {
            printf(", ");
        }
        printf("%.2f ", numbers[i]);
    }
    printf(" is: %.2f\n", avg);
}
```

Saving the file with name `average.c`. Then compiling the source file with

the following command:

```
$ clang -o average average.c
```

An executable binary file with name `average` will be created if everything goes right.

Output

Running the program with the following command:

```
$ ./average
Enter four numbers:
1 2 3 4
Average of 1.00 , 2.00 , 3.00 , and 4.00  is: 2.50
```

NOTE: All the programs are written in *Linux* environment. The executable binary files don't have an extension like .exe, .dmg, or .app beacuse in Linux whether a program is executable or not is determined by the permissions on the file, not the extension. And LLVM's *Clang* is used for the compilation of above source files which just happens to be my favourite.