ME-172
Computer Programming Language Sessional
Assignment No. 03

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

March 9, 2020

Problem 1

Write a program to evaluate the sine series using for loop.

Solution:

The sine series is given by the mathematical formula:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$$

The following program asks the user to enter the value of x and then it evaluates the sine series by adding up first 10 terms.

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


// Calculates the factorial of a non-negative integer.
long factorial(int n)
{
    long result = 1;
    for(; n > 1; n--)
    {
        result *= n;
    }
    return result;
}

int main(void)
{
    // Take user input for x
    float x;
    printf("Enter x: ");
    scanf("%f", &x);

    // Represents the sum of sine series up to n terms.
    double sum = 0;
```

```
        // Total number of terms from start to add up.
        const int n = 10;

        // Calculate
        for(int i = 0, j = 0; j < n; i+= 2, j++)
        {
            sum += pow(-1, j) * pow(x, i + 1) / factorial(i + 1);
        }

        // Show output
        printf("Sum of sine series up to 10th term = %lf\n", sum);
    }
```

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

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

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

Output

Running the executable with the following command:

```
$ ./sine
Enter x: 2
Sum of sine series up to 10th term = 0.909297
```

Problem 2

Write a program that determines the number of trailing zeros at the end of $x!$ (x factorial), where x is an arbitrary number. For instance, 5! is 120, so it has one trailing zero.

Solution:

The recursive definition of $x!$ is the following.

$$x! = x(x-1)!$$

And the iterative definition is the following. We will be using this definition in our code.

$$x! = x(x-1)(x-2)(x-3)...3.2.1$$

```
#include <stdio.h>
#include <stdbool.h>


// Calculates the factorial of a non-negative integer.
long factorial(int n)
{
    long result = 1;
    for(; n > 1; n--)
    {
        result *= n;
    }
    return result;
}

int main(void)
{
    // Take user input for x
    int x;
    printf("Enter x: ");
    scanf("%i", &x);

    long x_fact = factorial(x);
    long n = x_fact;

    // Represents the number of trailing zeros
    int count = 0;

    // Calculate
```

```
        for (;;)    // A while(true) would've been more appropriate
        {
            if (n % 10 == 0)
            {
                count++;
                n /= 10;
            }
            else
            {
                break;
            }
        }

        printf("%i! = %li\n", x, x_fact);
        printf("Total number of trailing zeros at the end of %i! = %i\n",
         x, count);
    }
```

Saving the file with name `trailingZeros.c`. Then compiling the source file with the following command:

```
$ clang -o trailingZeros trailingZeros.c
```

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

Output

Running the executable using the following command:

```
$ ./trailingZeros
Enter x: 10
10! = 3628800
Total number of trailing zeros at the end of 10! = 2
```

Problem 3

Draw a Pascal's triangle using C programming (for loop)

Solution:

The binomial theorem is given by the following formula:

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$$

If we just take the coefficients of $a^{n-k}b^k$ from the above binomial expansion formula and center-align them we get an interesting structure what we call a Pascal's Triangle. From combinatorics, the definition of $\binom{n}{k}$ is given by the following formula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

```
#include <stdio.h>


// Calculates the factorial of a non-negative integer.
long factorial(int n)
{
    long result = 1;
    for(; n > 1; n--)
    {
        result *= n;
    }
    return result;
}

// Prints the coefficients of nth row. (counting from 0th row)
void row(int n)
{
    for(int k = 0; k <= n; k++)
    {
        int coe = factorial(n) / (factorial(k) * factorial(n - k));
        printf("%i ", coe);
    }
    printf("\n");
}
```

```
int main(void)
{
    // Take user input for power of binomial, n.
    int n;
    printf("Enter n: ");
    scanf("%i", &n);

    // Print rows.
    for (int i = 0; i <= n; i++)
    {
        // Print preceding spaces
        for(int j = 0; j < n - i; j++)
        {
            printf(" ");
        }

        // Print row with height i.
        row(i);
    }
}
```

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

```
$ clang -o pascalsTriangle pascalsTriangle.c
```

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

Output

Running the executable with the following command:

```
$ ./pascalsTriangle
Enter n: 4
    1
   1 1
  1 2 1
```

```
 1 3 3 1
1 4 6 4 1
```

When a binomial raised to a power of n is expanded, we get a total of $n+1$ terms from the exansion. The first row in our output represents the coefficients of expansion of $(a+b)^0$, which is 1. And if we keep going up to (and including) $(a+b)^4$ we get the coefficients represented by the last row in our output. And there are $(4+1)=5$ coefficients (from $(4+1)=5$ terms) in the last row, just like we preached.

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.