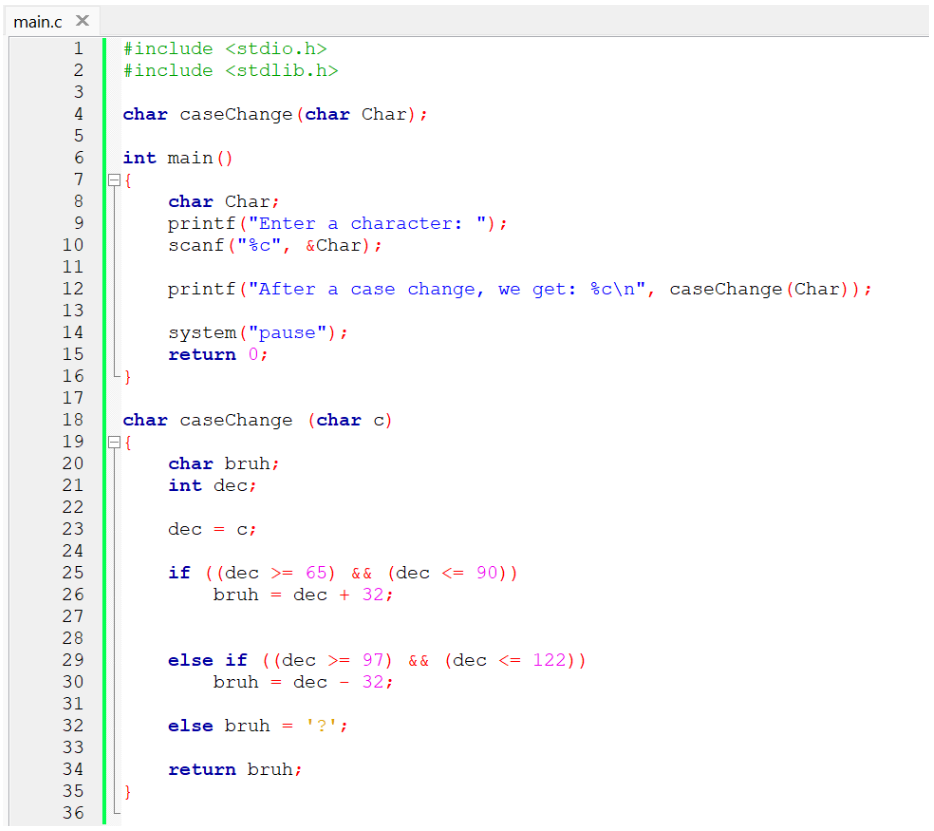
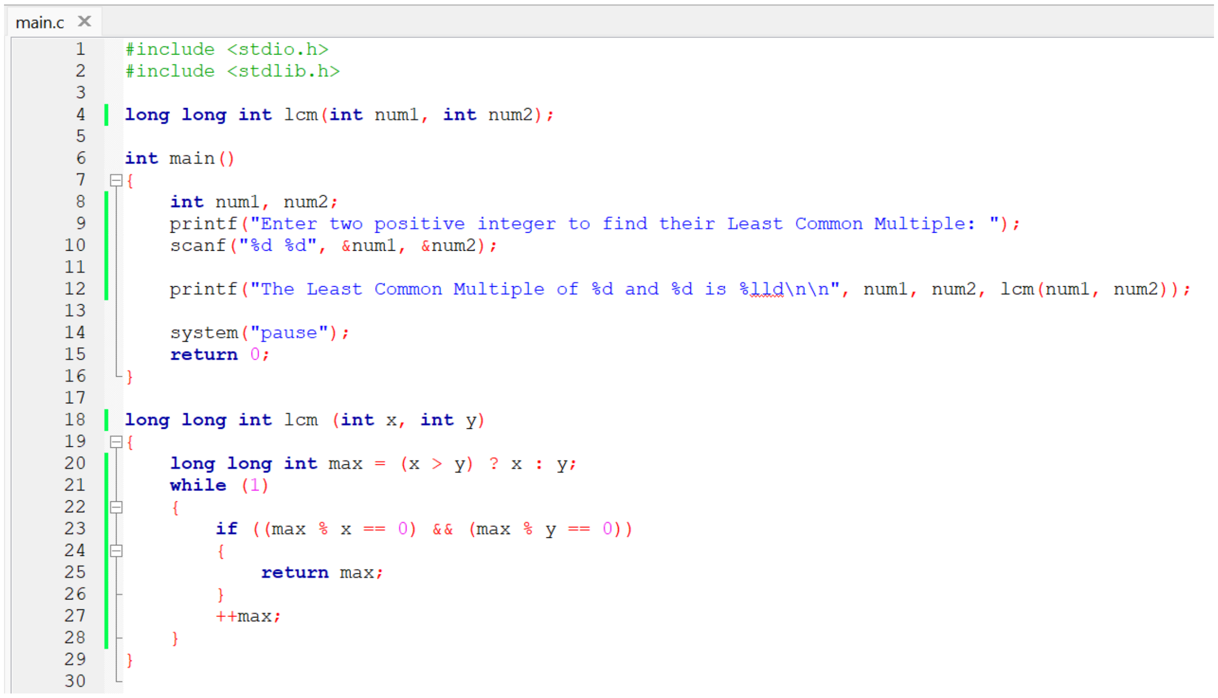
5.28



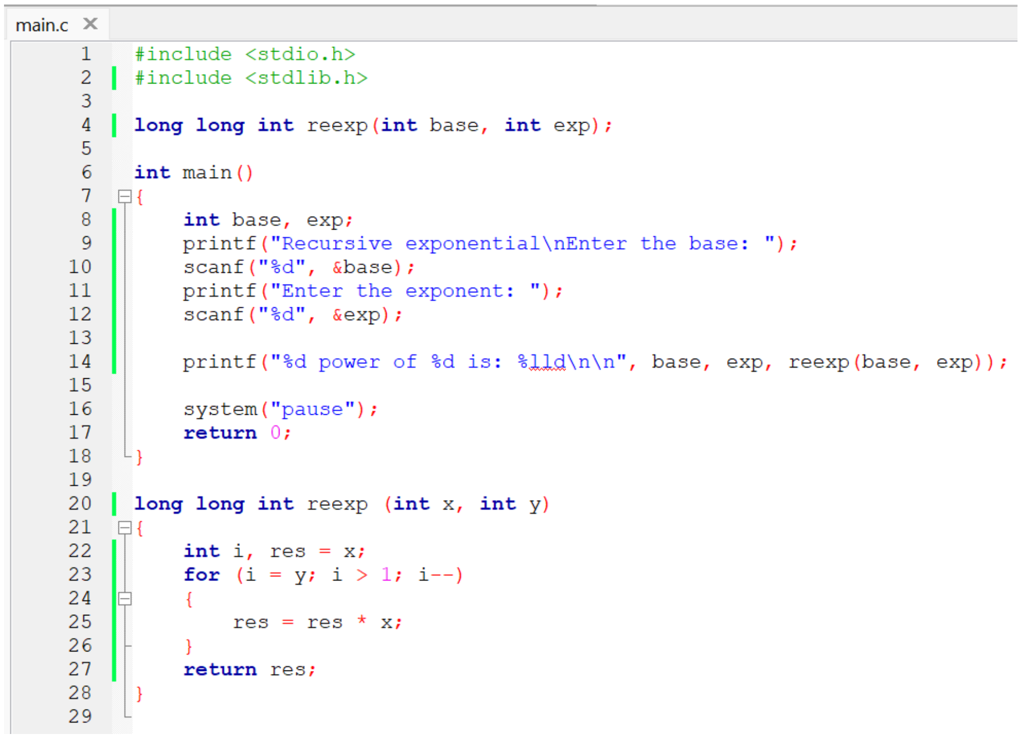
```c
#include <stdio.h>
#include <stdlib.h>

char caseChange(char Char);

int main()
{
    char Char;
    printf("Enter a character: ");
    scanf("%c", &Char);

    printf("After a case change, we get: %c\n", caseChange(Char));

    system("pause");
    return 0;
}

char caseChange (char c)
{
    char bruh;
    int dec;

    dec = c;

    if ((dec >= 65) && (dec <= 90))
        bruh = dec + 32;


    else if ((dec >= 97) && (dec <= 122))
        bruh = dec - 32;

    else bruh = '?';

    return bruh;
}

```

5.29



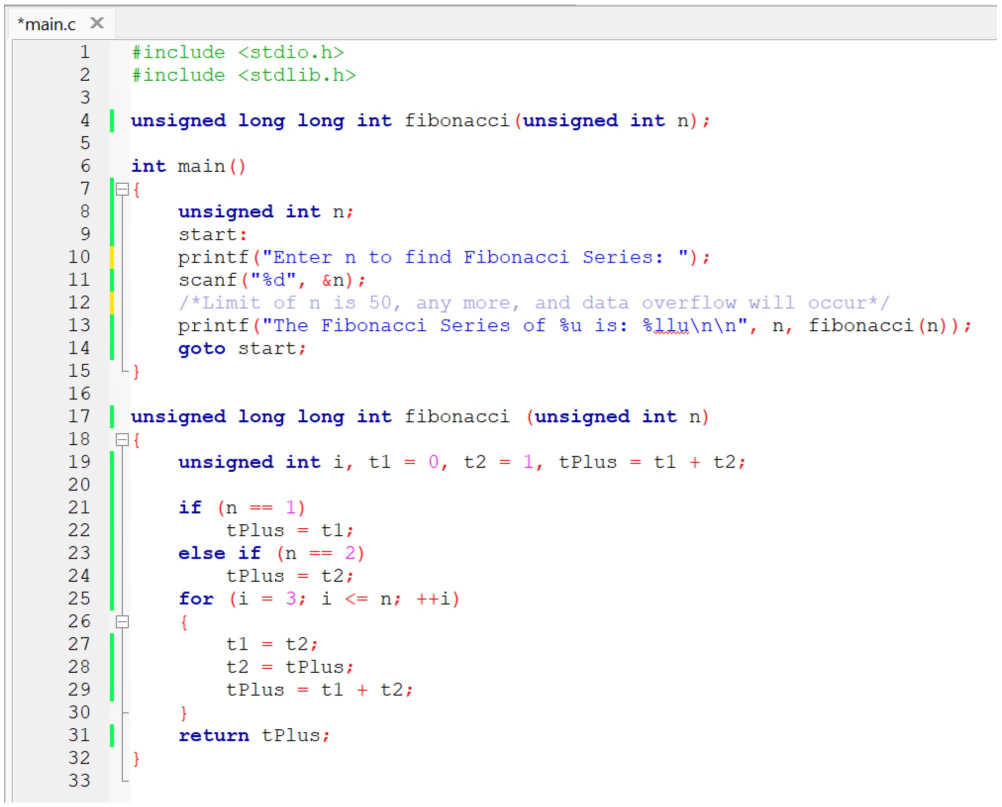
```c
#include <stdio.h>
#include <stdlib.h>

long long int lcm(int num1, int num2);

int main()
{
    int num1, num2;
    printf("Enter two positive integer to find their Least Common Multiple: ");
    scanf("%d %d", &num1, &num2);

    printf("The Least Common Multiple of %d and %d is %lld\n\n", num1, num2, lcm(num1, num2));

    system("pause");
    return 0;
}

long long int lcm (int x, int y)
{
    long long int max = (x > y) ? x : y;
    while (1)
    {
        if ((max % x == 0) && (max % y == 0))
        {
            return max;
        }
        ++max;
    }
}

```

5.34


main.c

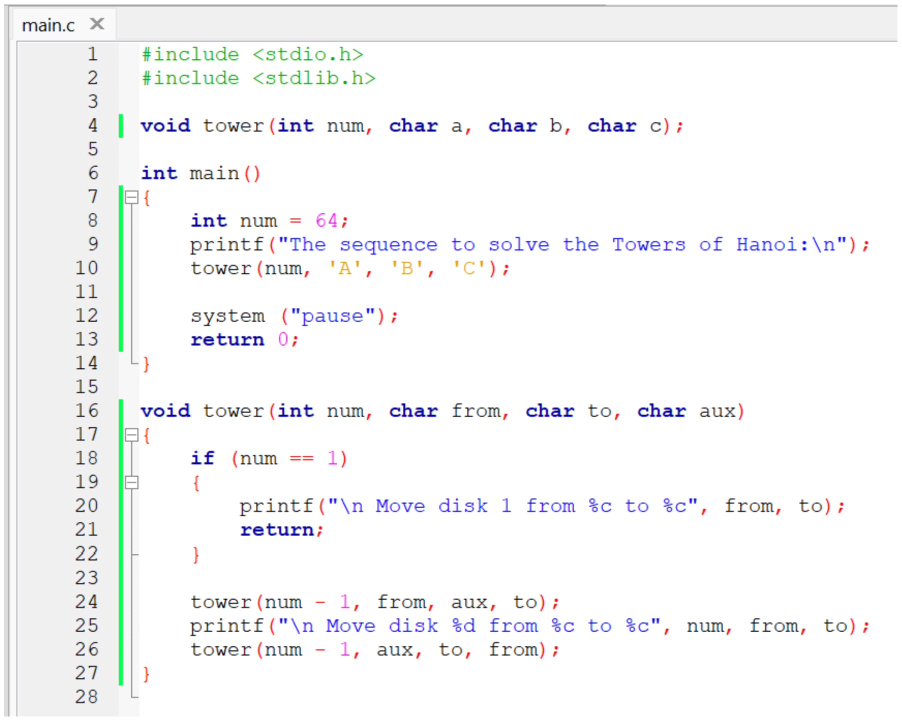
```c
#include <stdio.h>
#include <stdlib.h>

long long int reexp(int base, int exp);

int main()
{
    int base, exp;
    printf("Recursive exponential\nEnter the base: ");
    scanf("%d", &base);
    printf("Enter the exponent: ");
    scanf("%d", &exp);

    printf("%d power of %d is: %lld\n\n", base, exp, reexp(base, exp));

    system("pause");
    return 0;
}

long long int reexp (int x, int y)
{
    int i, res = x;
    for (i = y; i > 1; i--)
    {
        res = res * x;
    }
    return res;
}
```

5.35


*main.c

```c
#include <stdio.h>
#include <stdlib.h>

unsigned long long int fibonacci(unsigned int n);

int main()
{
    unsigned int n;
    start:
    printf("Enter n to find Fibonacci Series: ");
    scanf("%d", &n);
    /*Limit of n is 50, any more, and data overflow will occur*/
    printf("The Fibonacci Series of %u is: %llu\n\n", n, fibonacci(n));
    goto start;
}

unsigned long long int fibonacci (unsigned int n)
{
    unsigned int i, t1 = 0, t2 = 1, tPlus = t1 + t2;

    if (n == 1)
        tPlus = t1;
    else if (n == 2)
        tPlus = t2;
    for (i = 3; i <= n; ++i)
    {
        t1 = t2;
        t2 = tPlus;
        tPlus = t1 + t2;
    }
    return tPlus;
}
```

5.36



main.c

```c
#include <stdio.h>
#include <stdlib.h>

void tower(int num, char a, char b, char c);

int main()
{
    int num = 64;
    printf("The sequence to solve the Towers of Hanoi:\n");
    tower(num, 'A', 'B', 'C');

    system ("pause");
    return 0;
}

void tower(int num, char from, char to, char aux)
{
    if (num == 1)
    {
        printf("\n Move disk 1 from %c to %c", from, to);
        return;
    }

    tower(num - 1, from, aux, to);
    printf("\n Move disk %d from %c to %c", num, from, to);
    tower(num - 1, aux, to, from);
}

```

Conclusion:
Functions can be used to simplify a code, so it can be reused multiple times. The reusing of functions can easily be seen in number 5.36, where the function contains a function, which in turn triggers another function. The output from it is scarily fast, efficient, and mind boggling. I also learnt about how numerals work in programming, which uses the ASCII code, changing a decimal, hex, or any other form of bits into symbols and letters.

Code:
https://github.com/AldrichWijaya/Homework