Assignment 5: Functions

1. การทำงานของฟังก์ชัน Factorial

```
int factorial (int fac)
{
    int result = 1;
    for (int i = fac; i > 0;i--)
    {
        result = result * i;
    }
    return result;
}
```

ตัวอย่าง (input: 5)

เมื่อนำฟังก์ชัน factorial ไปรับค่า input ผ่านทางพารามิเตอร์ จะได้ว่าพารามิเตอร์ fac = 5

ฟังก์ชันจะประกาศตัวแปร result = 1 เพื่อเก็บค่าผลลัพธ์ที่ได้จากการคำนวณ จากนั้นเริ่มต้นลูป กำหนดค่า i เป็นตัวเลขในแต่ละลูป และเซ็ต i = fac ที่ได้มาคือ 5

ดังนั้นในตัวอย่างนี้ i = 5 และลดลงไปเรื่อย ๆ จนถึง 1

ในลูปแต่ละครั้งจะนำผลลัพธ์ที่เก็บไว้มาคูณกับ i จะได้สมการว่า result = result * i

นำมาเขียนเป็นตารางได้ดังนี้

i	แทนค่าในสมการ	ผลลัพธ์
5	1 * 5	5
4	5 * 4	20
3	20 * 3	60
2	60 * 2	120
1	120 * 1	120

หรือก็คือ 5 * 4 * 3 * 2 * 1 = 120

ผลลัพธ์ที่ได้คือ 120 และรีเทิร์นไป

Output: 120

2. ฟังก์ชัน Fibonacci

```
int fibo (int order)
{
    if (order <= 1)
        return order;
    else
        return fibo(order - 1) + fibo(order - 2);
}
```

ตัวอย่าง (input: 6)

เมื่อนำฟังก์ชัน fibo ไปรับค่า input ผ่านทางพารามิเตอร์ จะได้ว่าพารามิเตอร์ order = 6

นำค่า order มาตรวจสอบ ถ้าค่า order น้อยกว่าหรือเท่ากับ 1 ให้ส่งกลับค่า order ไป

แต่ถ้าค่า order มากกว่า 1 ให้รีเทิร์นค่าที่ได้จาก fibo(order - 1) + fibo(order - 2) ไป

จะได้ว่า

fibo(6) = fibo(6 - 1) + fibo(6 - 2) = fibo(5) + fibo(4)

fibo(5) = fibo(5 - 1) + fibo(5 – 2) = fibo(4) + fibo(3)

fibo(4) = fibo(4 - 1) + fibo(4 - 2) = fibo(3) + fibo(2)

fibo(3) = fibo(3 - 1) + fibo(3 - 2) = fibo(2) + fibo(1)

fibo(2) = fibo(2 - 1) + fibo(2 - 1) = fibo(1) + fibo(0)

fibo(1) = 1

fibo(0) = 0

เมื่อได้ค่า fibo(1) กับ fibo(0) รีเทิร์นค่ากลับมาจะได้ว่า

fibo(2) = fibo(1) + fibo(0) = 1 + 0 = 1

fibo(3) = fibo(2) + fibo(1) = 1 + 1 = 2

fibo(4) = fibo(3) + fibo(2) = 2 + 1 = 3

fibo(5) = fibo(4) + fibo(3) = 3 + 2 = 5

fibo(6) = fibo(5) – fibo(4) = 5 + 3 = 8

ถ้าเรียงเป็นลำดับจะได้ว่า 1, 1, 2, 3, 5, 8

ผลลัพธ์คือ 8 และรีเทิร์นไป

Output: 8

3. ฟังก์ชัน nCr

```
int nCr(int n, int r)
{
    return factorial(n) / (factorial(r) * factorial(n - r));
}
```

ตัวอย่าง (input1: 5, input2: 3)

เมื่อรับค่าพารามิเตอร์ถ้าใส่ n = input1 และ r = input2

จากสูตรทางคณิตศาสตร์ $\frac{n!}{r!(n-r)!}$ เราได้สร้างฟังก์ชัน factorial เอาไว้แล้ว จึงสามารถเอามาใช้งานในฟังชันนี้ได้ เขียนโปรแกรมจากสูตรได้ว่า factorial(n) / (factorial(r) * factorial(n-r))

เมื่อนำค่า n และ r ไปแทนจะได้ว่า

factorial(5) / (factorial(3) * factorial(5 - 3)) หรือก็คือ factorial(5) / (factorial(3) * factorial(2))

เมื่อฟังก์ชัน factorial ทั้งหมดคำนวณเสร็จและรีเทิร์นค่ากลับมาจะได้เป็น

120 / 6 * 2 = 120 / 12 = 10

ผลลัพธ์ที่ได้จากการคำนวณคือ 10 และรีเทิร์นกลับไป

Output: 10

4. ฟังก์ชัน nPr

```
int nPr(int n, int r)
{
    return factorial(n) / factorial(n - r);
}
```

ตัวอย่าง (input1: 5, input2: 3)

รับค่า input ผ่านพารามิเตอร์โดยใส่ input1 ใน n และ input2 ใน r

คำนวณหาค่า nPr จากสูตรคณิตศาสตร์ $\frac{n!}{(n-r)!}$

โดยใช้ฟังก์ชัน factorial ที่เราเขียนไว้แล้วมาใช้

แปลงจากสูตรมาเป็นโปรแกรมจะได้เป็น factorial(n) / factorial(n - r)

จาก input ที่เราได้มาจะกลายเป็น factorial(5) / factorial(5 - 3)

หรือก็คือ factorial(5)/ factorial(2)

หลังจากที่ฟังก์ชัน factorial คำนวณเสร็จและรีเทิร์นค่ากลับคืนมาจะได้ว่า

120 / 2 = 60

ผลลัพธ์จากการคำนวณคือ 60 และรีเทิร์นกลับไป

Output: 60

5. ฟังก์ชัน GCD

```
int GCD(int n, int r)
{
    if (n%r==0)
        return r;
    else return GCD(r, n%r);
}
```

ตัวอย่าง (input1: 18, input2: 81)

รับ input ผ่านทางพารามิเตอร์ โดยให้ input1 ผ่านตัวแปร n ให้ input2 ผ่านตัวแปร r

ฟังก์ชันนี้ทำงานโดยตรวจสอบว่าเศษที่ได้จาก n/r นั้นเท่ากับ 0 หรือไม่

ถ้าเศษเท่ากับ 0 จะส่งค่า r กลับไป

ถ้าเศษไม่เท่ากับ 0 จะเรียกฟังก์ชัน GCD โดยใส่พารามิเตอร์เป็น r กับเศษของ n/r หรือก็คือ GCD(r, n%r) นั้นเอง ซึ่ง n%r จะสลับค่าเองเมื่อ n < r

ในตัวอย่างนี้เราได้รับคือ 18 กับ 81 ซึ่งเมื่อหารหาเศษจะได้เศษ 9 ซึ่งมากกว่า 0 จึงเรียกใช้ฟังก์ชัน GCD(r, n%r) หรือก็คือ GCD(18, 9) ทำงานหาเศษที่ได้จากการหาร 18 ด้วย 9 คือ 0 ดังนั้นเมื่อเศษ 0 จึงรีเทิร์นค่า r หรือคือ 9 กลับมา

ผลลัพธ์ที่ได้จากการคำนวณคือ 9 และรีเทิร์นค่ากลับไป

Output: 9