main

```
const operations = require("./operations");

console.log(operations.add(50, 60)); //*Output: 110

console.log(operations.multiply(50, 60)); //* Output: 3000

console.log(operations.factorialize(4)); //* Output: 24

console.log(operations.toString("multiply", "th", 1000, 6)); //* Output: 6,000

console.log(operations.toString("factorialize", "th", 10)); //* Output: 3,628,800

operations.delay(() => {
    console.log("Done");
}, 1000); //* Desc: รอจนกว่าจะครบ 1 วินาที แล้วจึงจะ output ว่า Done ปล. จะไม่รอ และจะทำคำสั่งถัดไปทันที
```

operation

```
/**
 * หาผลบวกของตัวเลขที่รับมาทั้งหมด
 * @param  {...any} nums ชุดตัวเลขที่จะใช้คำนวณ
 * @returns {number} ผลรวมของตัวเลข
 */
const add = (...nums) => nums.reduce((prev, number) => prev + number, 0);

/**
 * หาผลของการคูณทั้งหมด
 * @param  {...any} nums ชุดตัวเลขที่จะใช้คำนวณ
 * @returns {number} ผลคูณของตัวเลขทั้งหมด
 */
const multiply = (...nums) => nums.reduce((prev, number) => prev * number, 1);

/**
 * หาผลรวมของ Factorial
 * @param {*} num ตัวเลขที่จะใช้หา
 * @returns {number} ตัวเลขของ Factorial
 */
const factorialize = (num) => {
   if (num < 0) return -1;
   else if (num == 0) return 1;
   else return num * factorialize(num - 1);
};

/**
```

```
 * สำหรับหน่วงเวลาการเรียกใช้งาน Function
 * @param {*} callbackFn Function ที่เราจะ execute
 * @param {*} ms เวลาที่รอ (หน่วย มิลลิวินาที)
 */
const delay = (callbackFn, ms = 1000) => {
   setTimeout(callbackFn, ms);
};

/**
 * เปลี่ยนจาก Number เป็น String ตามท้องถิ่นนั้นๆ (เช่นประเทศไทย จาก 6000 = 6,000)
 * @param {*} operation ตัวดำเนินการทางคณิตศาสตร์ (ที่มีอยู่ใน File นี้เท่านั้น) หรือเรากำหนดเอง
 * @param {*} locale code ตัวย่อประเทศ เช่น Thailand = th, Japan = jp
 * @param  {...any} nums ชุดตัวเลขที่ต้องการใช้
 * @returns {Function} เรียกใช้ Fuction toLocale เพื่อแปลง Nunmber เป็น String ตามท้องถิ่นนั้นๆ
 */
const toString = (operation, locale = "", ...nums) => {
   function toLocale() {
      switch (operation) {
         case "add":
            return add(...nums).toLocaleString(locale);
         case "multiply":
            return multiply(...nums).toLocaleString(locale);
         case "factorialize":
            return factorialize(nums[0]).toLocaleString(locale);
         default:
            return "Not operation found.";
      }
   }
```

```
    return toLocale(locale);
};

module.exports = {
    add,
    multiply,
    factorialize,
    toString,
    delay,
};
```