列表生成算法(不一定安全的列表)

生成一个列表，列表的每个元素都代表系统接收到的请求。
列表的每个元素都包括：所需要用到的三个资源的个数，任务几发起的请求，以及这是第几个请求。

1. **首先创建x个任务**，然后每个任务给定（a,b,c）个所需最大资源（这里其实就是创建一个3乘x的矩阵，也即银行家算法里的Max矩阵，然后矩阵里的每个值都是随机赋的整数值，注意别超过（小于等于）系统资源的上限（OSmax_a，OSmax_b，OSmax_c））。
2. **然后创建一个标志位列表**，这个列表的长度和任务数量（也就是x）相同，然后把列表初始化为全0。
3. **然后用一个循环生成请求列表**，开始循环每一次循环生成一个请求，然后把请求插到队列里，循环终止的条件为标志位列表全1，具体的逻辑为：
    - 复制一个Max矩阵为Need矩阵。
    - 在x个任务里随机抽取一个值n作为接下来我们要为哪个任务设定请求。
    - 最后读取这个任务的Need矩阵，获得这个请求的上限值（max_a，max_b，max_c）。
    - 如果max_a，max_b，max_c都为0，就让位于标志位列表的第n位加1，重新循环。
    - 否则，从max_a到0，max_b到0，max_c到0抽三个值，注意在抽三个值的时候，需要设置一个循环，如果抽到的值都是0，那么重抽。
    - 抽完之后，把所需要用到的三个资源的个数，任务几发起的请求，以及这是第几个请求（也就是这是第几次循环）存到列表里。
    - 然后在Need矩阵里删去对应的值，循环结束。

安全列表生成算法

参数为Need，Available，Allocation，作用是Need列表里如果说有全0的，那么对应的任务的资源归还给Available，并记录已经完成的tasks。

1. 一开始不初始化Max，request从0到OSmax取，然后OSmax减去request，重复这个过程，最后统计Max。
2. 这两个算法在列表生成文件里，生成完毕之后会把结果塞入txt文件。

银行家算法

输入参数：

- `x`（任务数）
- `resources`（请求所需的三种资源的个数，列表）
- `task`（请求属于第几个任务）
- `req_num`（是第几个请求）
- `Max矩阵`（一个列表，里面有x个列表，x个列表里每一个都装着三个数字，代表第x个任务的需要的最大资源数量）
- `Allocation矩阵`
- `Need矩阵`
- `Available列表`
- `Osmax列表`（OSmax_a, OSmax_b, OSmax_c）

银行家算法函数的的9个输入。

实现步骤

1. **创建函数实现银行家算法**，在上述基础上，一个循环调用银行家算法解决一个操作，直到处理完所有请求。
2. **具体操作**：
    1. 在进入循环之前，复制Max矩阵为Need矩阵，创建全0的Allocation矩阵，Available列表，然后Osmax列表复制给Available列表。
    2. 进入循环，循环每次运行一遍银行家算法函数，返回Max矩阵，Need矩阵，Allocation矩阵，Available列表。
    3. **银行家算法内部**：
        - 首先比较`resources`和`Need`里第`task`个任务，如果`resources`里三个值有任意一个比Need大，直接return。
        - 然后比较`Available`列表，如果`resources`里三个值有任意一个比Need大，直接return。
        - 创建`Max矩阵`，`Allocation矩阵`，`Need矩阵`，`Available列表`的副本`try_Max矩阵`，`try_Allocation矩阵`，`try_Need矩阵`，`try_Available列表`。
        - `try_Available列表`里的值加上`resources`，`try_Need列表`对应位置减去`resources`，`try_Allocation列表`对应位置加上`resources`。
        - 创建`work`列表并把`try_Available列表`赋给它。
        - **下面是一个循环**：
            - 在`try_Need矩阵`里找第一个小于等于`work`列表的列表。
            - 如果找到，更新对应的`try_Allocation列表`和`work`，并把对应的`try_Need`设置为不可操作（通过标记数组完成）。
            - 如果没有找到，函数return `x, task, req_num, Max, Allocation, Need, Available, false`。
            - 这个过程循环x次。
        - 循环完毕后，`Available列表`里的值减去`resources`，`Need列表`对应位置减去`resources`，`Available列表`里的值加上`resources`。
        - return `x, task, req_num, Max, Allocation, Need, Available, true`。