1. (5 points)There exist two processes P_1 and P_2 in a multiprogramming batch system. P_2 enters into the system 10ms later than P_1, and their executing traces, i.e. alternating sequences of CPU bursts and I/O bursts, are as follows:

P_1: computing, 80ms → I/O operation, 100ms → computing, 40ms

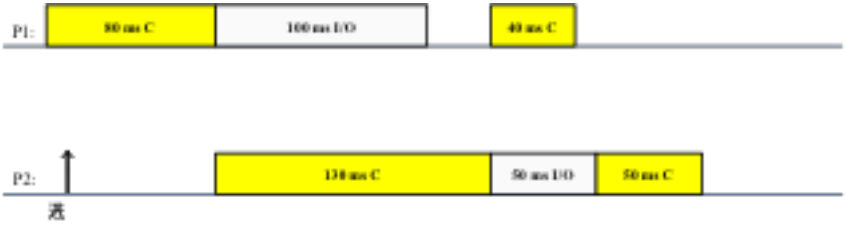
P_2: computing, 130ms → I/O operation, 50ms → computing, 50ms

It is assumed that time costs of CPU scheduling and process switch are omitted, what is the maximal throughput for completing these two processes, and why?

Answers：

1. 最大吞吐量 = 2/(80+130+50+50) = 2/310= **1/155** 个/ms (3 points)

2. 当**P_1**、**P_2**的CPU计算和I/O操作最大程度并行执行时，2个进程总的执行时间最短，系统吞吐量达到最大，如下图所示；（2 points）



2. In a computer system, the users submit to the system their computational tasks as jobs, and all these jobs are then stored as the standby jobs on the disk.

The job scheduler (also known as *long-term scheduler*) selects the standby jobs on the disk, creates new processes in memory for them, and then starts executing of these processes. Each job's ID is the same as that of the process created for it, for example, J_i and P_i.

When the number of concurrent processes in memory is lower than *three*, the job scheduler takes the FCFS algorithm to select a standby job on the disk to create a new process. Otherwise, the jobs should wait on the disk.

For the processes in memory, the process scheduler (also known as *short-term scheduler*) takes the non-preemptive priority-based algorithm to select a process and allocates the CPU to it.

It is assumed the system costs resulting from job and process scheduling are omitted.

Consider the following set of Jobs J_1, J_2, J_3 , J_4 and J_5. For $1 \le i \le 5$, the arrival time of each J_i , the length of the CPU burst time of each process P_i, and the priority number for each J_i/P_i are given as below, and a smaller priority number implies a higher priority.

Job	Arrival Time	Burst Time	Priority Number

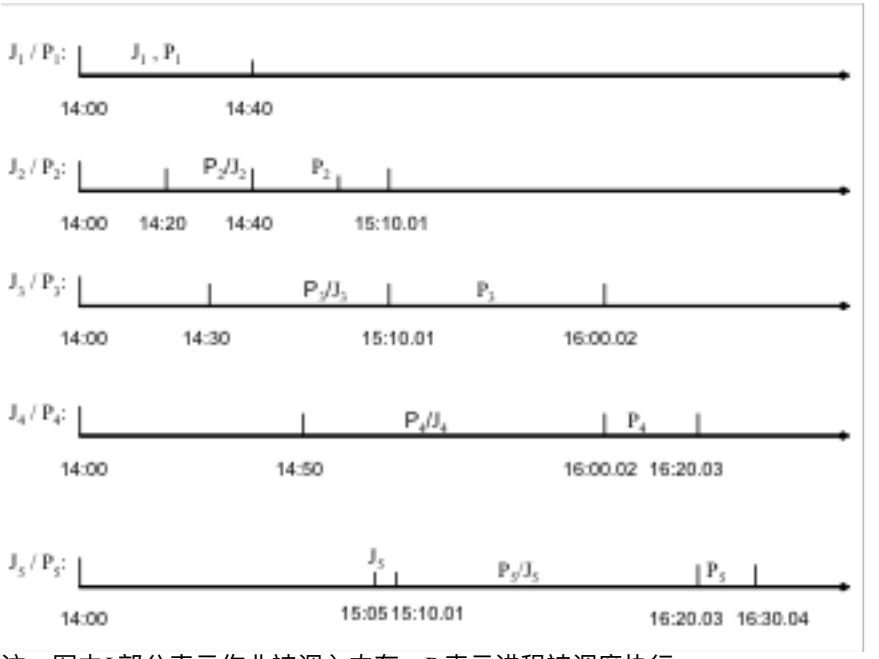
		(minute)	
J_1	14:00	40	4
J_2	14:20	30.01	2
J_3	14:30	50.01	3
J_4	14:50	20.01	5
J_5	15:05	10.01	5

(1) **Illustrate the execution of each job/process by charts.**
(2) **What is the turnaround time of each job?**
(3) **What is the waiting time of each job?**

Note: The waiting time of a job includes the time it waits on the disk and that it (as a process) waits in memory.

Answer:

(1)



注：图中J_i部分表示作业被调入内存，P_i表示进程被调度执行。

J_1到达时，内存中并发进程数=0<3，作业被直接调入内存，创建进程P_1；P_1被调度程序选中，开始执行。

在P_1执行过程中，J_2到达，内存中并发进程数=1<3，作业被直接调入

内存，创建进程P_2；此时P_1在执行，由于采用非抢占式进程调度，P_2处于就绪等待状态。

在P_1执行过程中，J_3到达，内存中并发进程数=2<3，作业被直接调入内存，创建进程P_3，等待P_2执行完毕。

P_1结束后，作业J_1随之结束，系统内有作业J_2、J_3对对应的进程P_2、P_3。进程调度选择高优先级的P_2开始执行。

P_2执行过程中，J_4到达时，内存中并发进程数=2<3，作业被直接调入内存，创建进程P_4，等待P_2执行完毕。

P_2执行过程中，J_5到达时，内存中并发进程数=3，必须等到P_2结束后，系统内并发进程数<3，方能创建进程P_5。

P_2执行完毕后，系统内有2个作业J_3、J_4对应的进程P_3、P_4，内存中并发进程数=2<3。此时，长期调度程序首先为作业J_5创建进程P_5。然后，进程调度程序从P_3、P_4、P_5中，选择高优先级的P3开始执行。

P_3执行完毕后，系统内有2个作业J_4、J_5对应的进程P_4、P_5，2者的优先级相同，按照FCFS原则，P_4先执行。

P_4执行完毕后，剩余的唯一进程P_5开始执行。

(2) J_1： $T_1 = 40$ (min)
J_2： $T_2 = 20 + 30.01 = 50.01$ (min)
J_3： $T_3 = 40.01 + 50.01 = 90.02$ (min)
J_4： $T_4 = 70.02 + 20.01 = 90.03$ (min)
J_5： $T_5 = 5.01 + 70.02 + 10.01 = 85.04$ (min)

(3) J_1： $W_1 = 0$ (min)
J_2： $W_2 = 20$ (min)
J_3： $W_3 = 40.01$ (min)
J_4： $W_4 = 70.02$ (min)
J_5： $W_5 = 75.03$ (min)

3. 某多道程序设计系统供用户使用的主存为100K，磁带机2台，打印机1台。采用可变分区内存管理，采用静态方式分配外围设备（以先申请先满足、非抢占方式分配磁带机、打印机），忽略作业I/O时间。现有作业序列如下：

作业号	进入输入	运行时间	主存需求量	磁带机需求量	打印机需求量

	井时间				
1	8:00	25分钟	15K	1	1
2	8:20	10分钟	30K	0	1
3	8:20	20分钟	60K	1	0
4	8:30	20分钟	20K	1	0
5	8:35	15分钟	10K	1	1

作业调度采用FCFS策略，优先分配主存低地址区，且不准移动已在主存的作业，**在主存中的各作业平分CPU时间（时间片轮转法）。**

现求：

(1) 作业被调度的先后顺序

(2) 全部作业运行结束的时间

(3) 作业平均周转时间

(4) 最大作业周转时间

答案：

本题综合了作业调度、进程调度、对外设竞争、对主存竞争。

(1) 作业调度选择的作业顺序为：作业1、作业3、作业4、作业2、作业5

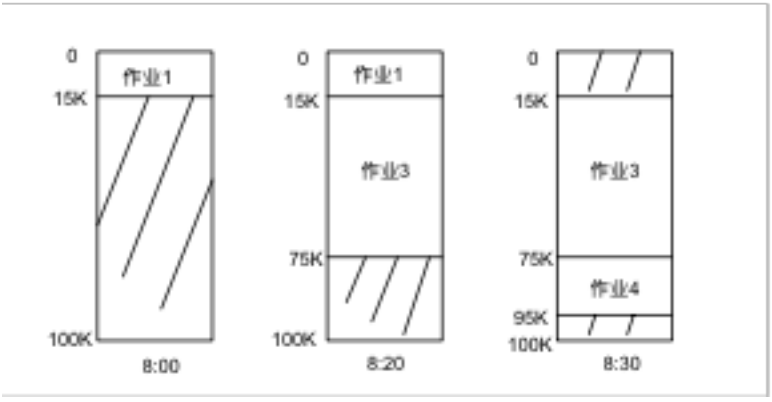
(2) 全部作业运行结束的时间为9:30

(3) 周转时间：作业1为30分钟，作业2为55分钟，作业3为40分钟，作业4为40分钟，作业5为55分钟

(4) 平均作业周转时间为44分钟

(5) 最大作业周转时间为55分钟

调度过程描述：



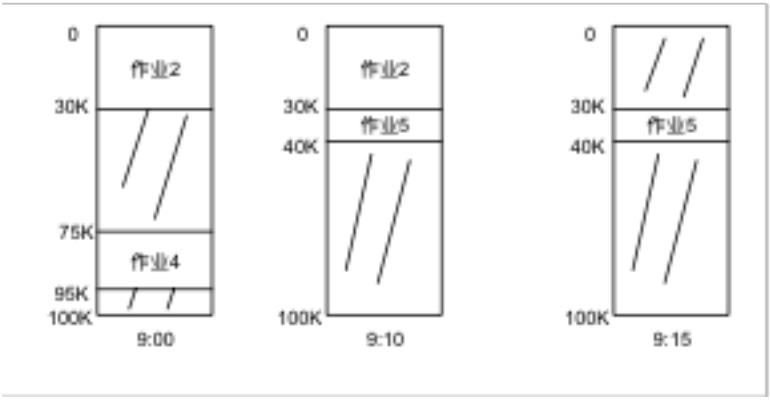
8:00 作业1到达，占有资源（内存(15)、磁带机(1)、打印机(1)），并调入内存运行；

8:20 作业2、3同时到达，但作业2因为分不到打印机—打印机资源受限，只能在后备队列中等待；

作业3所需资源（内存(60)，磁带机(1)）满足，调入内存运行；

此时，作业1还需5分钟CPU时间，但由于采用时间片分时调度，作业3与作业1平分CPU时间，作业1还需运行10分钟；

8:30 作业1在8:30结束，释放磁带机与打印机。但作业2仍不能执行：因为分配给作业3的内存无法压缩、移动，在当前主存内，低端15K空闲区域，高端25K区域，无法满足作业2的30K的要求。
8:30 作业4到达，其所需资源（内存(20)，磁带机(1)）满足，被调入内存执行，与作业3分享CPU。
8:35 作业5到达，资源需求（内存(10)、磁带机(1)、打印机(1)），因为分不到打印机、磁带机，只能在后备队列等待。



9:00 作业3运行结束（为什么？5*2+15*2），释放磁带机。此时，作业2的主存、打印机需求均可满足，作业2投入运行；作业5到达时间晚，只能等待。
9:10 作业4运行结束，作业5因为仍然分不到打印机，只能在后备队列中等待。
9:15 作业2运行结束，作业5投入运行
9:30 全部作业执行结束。

4. Considering a real-time system, in which there are 4 real-time processes P_1, P_2, P_3 and P_4 that are aimed to react to 4 critical environmental events e_1, e_2, e_3 and e_4 in time respectively.

The arrival time of each event e_i , $1 \le i \le 4$, (that is, the arrival time of the process P_i), the length of the burst time of each process P_i, and the deadline for each event e_i are given below. Here, the deadline for e_i is defined as the absolute time point before which the process P_i must be completed.

The priority for each event e_i (also for P_i) is also given, and a smaller priority number implies a higher priority.

Events	Process	Arrival Time	Burst Time	Priorities	Deadline
e_1	P_1	0.00	4.00	3	7.00
e_2	P_2	3.00	2.00	1	5.50
e_3	P_3	4.00	2.00	4	12.01
e_4	P_4	6.00	4.00	2	11.00

(1) **Suppose that priority-based preemptive scheduling is employed,**

a) **Draw a Gantt chart illustrating the execution of these processes**

b) **What are the average waiting time and the average turnaround time**

c) **Which event will be treated with in time ?**

(2) **Suppose that FCFS scheduling is employed,**

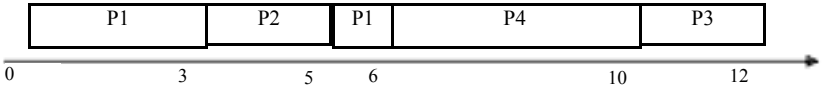
a) **Draw a Gantt chart illustrating the execution of these processes**

b) **What are the average waiting time and the average turnaround time**

c) **Which event will be treated with in time, that is, the process reacting to this event will be completed before its deadline?**

Answers:

(1) 甘特图如下

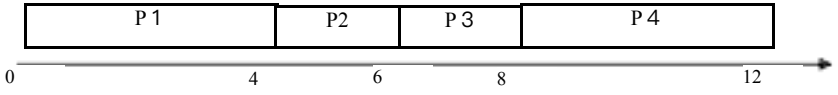


平均等待时间=[(5—3) + (3—3) + (10—4) + (6—6)] /4 = 2

平均周转时间=[(6—0) + (5—3) + (12—4) + (10—6)] /4 = 5

根据各进程的完成时间点和所对应的事件的deadline可知，全部 4 个事件均可得到及时响应.

(2) 甘特图如下



平均等待时间=[(0—0) + (4—3) + (6—4) + (8—6)] /4 = 1.25

平均周转时间=[(4—0) + (6—3) + (8—4) + (12—6)] /4 = 4.25

根据各进程的完成时间点和所对应的事件的deadline可知，事件e1和e3可得到及时响应.

5. As illustrated in the figure, on the two sides of a one-plank bridge(独木桥), there are two groups of soldiers that are composed of *m* and *n* people respectively and need to cross the bridge, but the narrow bridge allows only one group of the soldiers in the same direction to cross at the same time. One group of the soldiers is permitted to cross as long as there are no people on the bridge. Once one group of the soldiers begins walking on the bridge, the other group should be waiting to start crossing until all members of the first group have passed the bridge.

Please design two semaphore-based processes to describe the crossing actions of the soldiers in the two groups. It is required

(1) to define the semaphores and variables needed, explain their roles?, and give their initial values; and

(2) to illustrate the structures of processes for the soldiers in each group.

Answers:

该问题可归结为读写者问题，但2队士兵均为读者。

（1）定义2个整型变量count1、count2，分别用于计数每组士兵过桥人数。
定义2个二元信号量mutex1、mutex2，用于控制对count1、count2的互斥访问。
定义二元信号量bridge，用于2队士兵间的互斥。

初始化：
count1=0，count2=0；
mutex1=1、mutex2=1；
bridge=1；

（2）

```
P1()
{
  wait(mutex1);
  count1 ++;
  if count1 = =1
    wait(bridge);  // 该组第1个士兵过桥时，阻塞另一组过桥
  signal(mutex1);
     …….
  crossing the bridge;
```

```
    ……
  wait(mutex1);
  count1 --;
  if count1 = =0
    signal(bridge);  // 该组最后1个士兵过桥后，允许另一组过桥
  signal(mutex1);
}.


P2()
{
  wait(mutex2);
  count2 ++;
  if count2 = =1
    wait(bridge);  // 该组第1个士兵过桥时，阻塞另一组过桥
  signal(mutex2);
    …….
  crossing the bridge;
    ……
  wait(mutex2);
  count2 --;
  if count2 = =0
    signal(bridge);  // 该组最后1个士兵过桥后，允许另一组过桥
  signal(mutex2);
}.
```

6. Consider the *bounded-buffer (also known as producer-consumer problem)*. There are several producer processes and consumer processes, and these processes share a pool of buffer. The pool of buffer consists of *n* buffers, and each buffer can hold only one item. The producer process produces one item and puts it into an empty buffer in the pool. The consumer process takes one item from a full buffer in the pool and consumes this item.

At each moment, *at most one producer process and at most one consumer process* are permitted to simultaneously operate on the buffer pool, i.e.,

- **one producer process is permitted to access the buffer lonely**
- **one consumer process is permitted to access the buffer lonely**
- **if two or more processes want to access the buffer simultaneously,**
  - **two or more *producer processes* are not permitted to simultaneously operate on the buffer, they should access the buffer mutual exclusively**
  - **two or more *consumer processes* are also not permitted to simultaneously operate on the buffer, they should access the buffer mutual exclusively**
  - **only one *producer process* and one *consumer process* are allowed to simultaneously operate on different items in the buffer**

Write an algorithm to synchronize producer processes and consumer processes by using semaphores.

Answer1:

(1) Principles:

define the semaphores ***empty*** and ***full*** to count the number of empty buffers and full buffers in the pool, respectively;

define binary semaphore ***mutex1*** to control all ***producers*** to access the empty buffers in the pool mutual exclusively;

define binary semaphore ***mutex2*** to control all ***consumers*** to access the full buffers in the pool mutual exclusively;

(2) Algorithm

Semaphores empty, full

Binary semaphores mutex1, mutex2

Initially,

empty:= n; full:=0; mutex1:=1; mutex2:=2;

The structure of the *producer* process is as follows:

```
do {  …

    produce an item in nextp
        …

     wait(empty);
     wait(mutex1);
     search for an empty buffer
        …
     add nextp to the empty buffer
        …

   signal(mutex1);
     signal(full);
  } while (1);
```

The structure of the ***consumer*** process is as follows:

```
do {  …

     wait(full);
     wait(mutex2);
      search for an full buffer
        …
     remove item in buffer[j] to next ;

     ….
```

```
        signal(mutex2);
        signal(empty);
         …
        consume the item in nextc

        …
    } while (1);
```

Answer2:

Semaphores empty, full;
Binary semaphores mutex1, mutex2, mutex;
Array Flag[n]; /* Flag[i] =0 indicates that buffer[i] is empty,
Flag[i] =1 indicates that buffer[i] is full, $0 \le i \le n-1$; */
Initially,
empty:= n; full:=0; mutex1:=1; mutex2:=1; mutex=1;
Flag[i] =0, $0 \le i \le n-1$;

The structure of the *producer* process is as follows:

```
do {  …

        produce an item in nextp
            …

        wait(empty);
        wait(mutex1);

        wait(mutex);
        search on array Flag for an empty buffer[i], where Flag[i] =0;
        signal(mutex);
            …
        add nextp to the empty buffer[i];

        wait(mutex);
        Flag[i] : = 1;
        signal(mutex);

      signal(mutex1);
        signal(full);
```

…
} while (1);

The structure of the ***consumer*** process is as follows:

do { …

wait(full);
wait(mutex2);

wait(mutex);
search on array ***Flag*** for an full buffer[j], where Flag[j] =1;
signal(mutex);
…
remove item in buffer[j] to *next* ;

wait(mutex);
Flag[j] : = 0;
signal(mutex);

signal(mutex2);
signal(empty);
…
consume the item in *nextc*

…
} while (1);

7. 有1个仓库，可以存放X、Y两种产品，仓库容量足够大，但要求：
1）每次只能存入一种产品X或Y，
2）满足 —N < X产品数量 — Y产品数量 < M，M、N为正整数。

用信号量实现产品X、Y的入库过程。

Answers：
产品数量制约条件：

X产品数量 — Y产品数量< M
X产品数量 — Y产品数量 > —N

设置2个信号量控制X、Y的数量：

1） sx 表示当前允许X产品比Y产品多入库的数量，即在当前库存量和Y产品不入库的情况下，还可以允许sx个X产品入库；初始时，若不放Y而仅放X产品，则sx最多为M—1个。
2） sy 表示当前允许Y产品比X产品多入库的数量，即在当前库存量和X产品不入库的情况下，还可以允许sy个Y产品入库；初始时，若不放X而仅放Y产品，则sy最多为N—1个。

当往库中放入一个X产品时，则允许存入Y产品的数量加1，故信号量sy应加1；
当往库中放入一个Y产品时，则允许存入X产品的数量加1，故信号量sx应加1；

Mutex： semaphore=1 互斥信号量
sx, sy: semaphore
sx=M-1, sy=N-1

```
Process X
  Repeat
   Wait(sx)
   Wait(mutex)
   将X入库
   Signal(mutex)
   Signal(sy)
   Until false
Process Y
  Repeat
```

Wait(sy)
Wait(mutex)
将Y入库
Signal(mutex)
Signal(sx)
Until false

8. Here are three workers W_1, W_2 and W_3. W_1 and W_2 are responsible for producing the part(零件) P_1 and part P_2 respectively, and W_3 takes charge of assembling (装配) the part P_3.

W_1 produces one P_1 and then puts it into the cargo tank(货箱) T_1; W_2 manufactures one P_2 and then puts it into the cargo tank T_2. W_3 takes P_1 from T_1 and P_2 from T_2, and then assembles P_1 and P_2 into P_3.

It is assumed that (1) T_1 can hold 8 parts of P_1, and T_2 can hold 10 parts of P_2, and they are all initially empty; (2) each time W_3 can takes only one part of P_1 from T_1 and one part of P_2 from T_2, and then assembles one part of P_3; (3) T_1 and T_2 are permitted only to be taken or put in the mutually exclusive mode, that is, at any time only one worker are allowed to operate on T_1 or T_2.

Please design three semaphore-based programs for W_1, W_2 and W_3, to describe their production activities. It is required

(1) to define the semaphores for synchronizing the three processes, explain the role of each semaphore, and give their initial values; and

(2) to illustrate the structures of processes for W_1, W_2 and W_3.

Answers:

(1)

信号量定义：

Binary semaphore mutex1, mutex2

Semaphore empty1, full1, empty2, full2

信号量含义：

mutex1用于控制W_1和W_2对T1的互斥访问，mutex2用于控制W_1和W3对T_2的互斥访问；

empty1和empty2分别表示T_1和T_2中的空闲容量，full1和full2分别表示T_1和T_2中已经被占用的容量。

信号量初值：

mutex1=1; mutex2=1;

empty1=8; empty2=10;

full1=0; full2=0;

(2)

W_1：

Produce one P_1 ;
Wait(empty1);
Wait(mutex1);
Puts it into T_1,
Signal(mutex1);
Signal(full1);

W_2:
Produce one P_2 ;
Wait(empty2);
Wait(mutex2);
Puts it into T_2,
Signal(mutex2);
Signal(full2);

W_3:
Wait(full1);
Wait(mutex1);
takes one P_1 from T_1;
Signal(mutex1);
Signal(empty1);

Wait(full2);
Wait(mutex2);
takes one P_2 from T_2;
Signal(mutex2);
Signal(empty2);

Assembles P_1 and P_2 into P_3.

9. In a data collection system, there are a data collection process, a data manipulation process, and a data output process. The Data collection process puts one unit of data collected into Buffer 1, the data manipulation process gets one unit of data from Buffer 1 to manipulate and puts the results into Buffer 2, and then the data output process gets one unit of data from Buffer 2, and outputs them. It is assumed that Buffer 1 can keep three units of data and Buffer 2 can keep two units of data at most.

1) Define the semaphores used to synchronize the three processes, and set their initial values.

2) Please design three processes for the data collection process, the data manipulation process, and the data output process respectively by using semaphores defined in 1).

Answers:

(1) (4 points)

```
binary semaphore  mutex1, mutex2
semaphore full1, full2, empty1, empty2
mutex1=1; mutex2=1;
empty1=3; full1=0;  empty2=2; full2=0;
```

(2) (4 3=12 poin ts)

a. the collection process:

```
collect one unit of data;
wait(empty1);
wait(mutex1);
 put the data into buffer1;
signal(mutex1);
signal(full1);
```

b. the manipulating process:

```
wait(full1);
wait(mutex1);
  remove one unit of data from buffer1;
signal(mutex1);
signal(empty1);
manipulate the data;
wait(empty2);
wait(mutex2);
 put the data into buffer2;
signal(mutex2);
signal(full2);
```

c. the output process:

```
wait(full2);
wait(mutex2);
  remove one unit of data from buffer2;
signal(mutex2);
signal(empty2);
output the data;
```

30% deadlock

10. A computer system has the main memory of size 150MB, which has been allotted to three processes P_1 , P_2 and P_3. As shown in below, each process has been allocated some size of main memory, and the maximum size of main memory needed by each process is also given.

It is assumed that: (1) the main memory having been allocated to a process cannot be preempted by the other processes; (2) swapping in and swapping out of processes are not allowed.

Process	*Maximum Memory Needed*	*Memory Allocated*
***P*1**	**70MB**	**45MB**
***P*2**	**60MB**	**40MB**
***P*3**	**60MB**	**15MB**

When another process *P*4 enters into the system, it immediately requests main memory of the size 25MB, and its maximum size of main memory needed is 60MB.

Answer the following questions, by means of the banker's algorithm:

(1) **What are the contents of the matrix *Allocation*, *Max*, *Need* and the vector *Available*?**

(2) **Can the *P*4's request of main memory be granted immediately? Will be the system in a safe state? and why?**

Answers:

1)

Process	*Max*	*Allocated*	*Need*	*Available*
*P*1	70MB	45MB	25	25
*P*2	60MB	40MB	20	
*P*3	60MB	15MB	45	
*P*4	60MB	25MB	35	

Or

Process	*Max*	*Allocated*	*Need*	*Available*
*P*1	70MB	45MB	25	50
*P*2	60MB	40MB	20	
*P*3	60MB	15MB	45	
*P*4	60MB	0	60	

2)

It can be granted immediately.

In a safe state.

Have safe sequence ***P*1 *P*3 *P*4 *P*2**

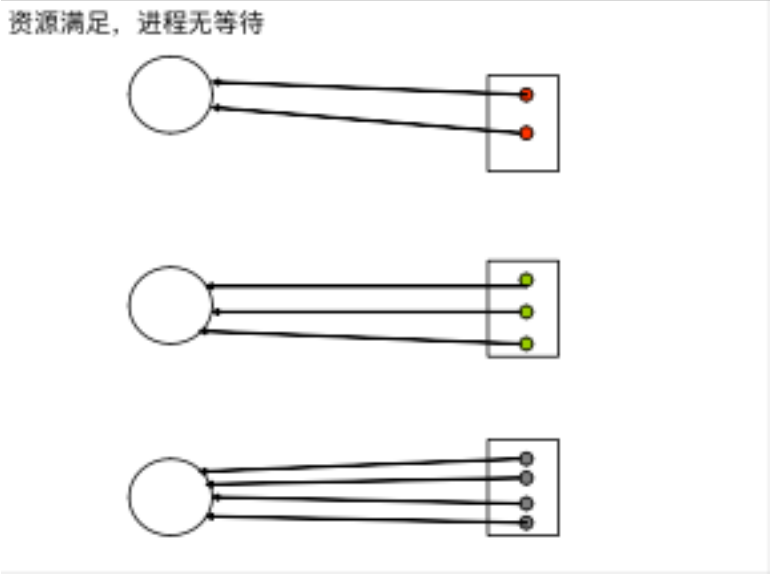
11. 某系统有n台互斥使用的同类设备，3个并发进程分别需要使用2、3、4台设备，可确保系统发生死锁的设备数目N最大为多少?

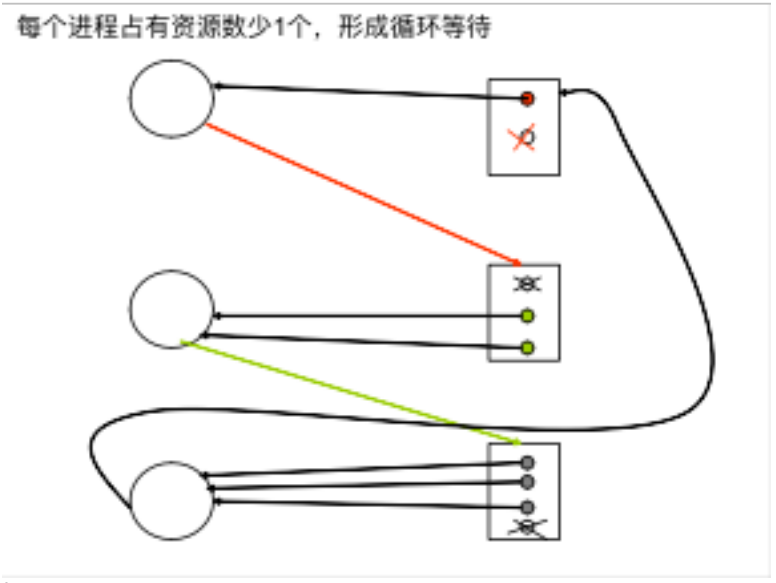
答案：

资源/设备总实例数N = 进程资源需求总数L — 进程总数m*1

N= (2+3+4) — 3*1

= 6




每个进程占有资源数少1个，形成循环等待