1. 今有三个并发进程R 、M、P,它们共享一个缓冲区，三个进程的执行顺序是R先从输入设备读信息，每读一个记录，把它存放在缓冲区，然后M在缓冲区加工读入的记录，M加工完成最后由P把记录打印输出，这时又可以从头运行R向缓冲区存放下一个记录，如此往复(前趋图如下所示)。试在下述类PASCAL程序中空白处填上信号量初值、Wait、Singal操作实现三个进程正确的并发执行。

R M P

var a,b, c :semaphore:=**1，0，0**;

begin

parbegin

R: begin

repeat

Read from i/o

**﹎﹎﹎Wait(a)﹎﹎﹎**;

Add message to buffer;

**﹎﹎﹎Signal(b)﹎﹎﹎**;

until

end

M: begin

repeat

**﹎﹎﹎Wait(b)﹎﹎﹎**;

Get message from buffer;

Process message;

Add message to buffer;

**﹎﹎﹎Signal(c)﹎﹎﹎**;

until

end

P: begin

repeat

**﹎﹎﹎Wait(c)﹎﹎﹎**;

Get message from buffer;

**﹎﹎﹎Signal(a)﹎﹎﹎**;

Print message;

until

end

parend

end

1. 仓库中有10个货架（即最多可装下10件货物），只有一张登记表，欲存入的货物须先在登记表上登记，然后存入；每存入5件货物，即通知出货员出货，并在出货时注销登记信息。

解：

var mutex, seat, shipment: semaphore := ① **1, 10, 0**  ; /*定义变量*

count, i: integer = 0, 0;

procedure stockin; *//存货进程*

begin

② **wait(seat)**  ;

③ **wait(mutex)** ;

登记一件货物，并将其存入仓库……;

count := count + 1;

if (count == 5) then

begin

⑤ **signal(shipment)**  ;

count := 0;

end;

④ **signal(mutex)**  ;

end;

procedure stockout; //*出货进程*

begin

⑥ **wait(shipment)**  ;

⑦ **wait(mutex)** ;

十件货物出仓，并撤消它们的登记信息….;

⑧ **signal(mutex)**  ;

for (i=0; i<5; i++) ⑨ **signal(seat)**  ;

end;

1. 有两个并发进程read和print共享缓冲器B。进程read负责从输入设备上读信息，每读出一个记录后把它存放到缓冲器B中，进程print将B中的记录取出并打印输出。缓冲器B每次只能存放一个记录。要求两个进程协调完成任务，使打印出来的与读入的记录的个数，次序完全一样。请用wait, signal操作，写出它们的并发程序。

var mutex, empty, full: semaphore := ① **1, 1, 0**

B, rRecord, wRecord: item;

procedure read;

begin

repeat

读一个记录rRecord;

② **wait(empty);**

③ **wait(mutex);**

B : = rRecord;

④ **signal(mutex);**

⑤ **signal(full);**

until false;

end;

procedure print;

begin

repeat

② **wait(full);**

③ **wait(mutex);**

wRecord := B;

④ **signal(mutex);**

⑤ **signal(empty);**

打印一个记录wRecord;

until false;

end;

1. 某车站售票厅，任何时刻最多可容纳30名购票者进入，当售票厅中少于30名购票者时，则厅外的购票者可立即进入，否则需在外面等待。若把一个购票者看作一个进程，用信号量buyer同步这些并发进程，其算法如下：

var buyer: semaphore := ①  **30** ; //*定义购票者信号量*

procedure wicket *//售票进程*

begin

②  **wait(buyer)** ;

购票者进入售票厅；

购票；

退出；

③  **signal(buyer)** ;

end;

buyer >0，buyer的值表示④  **可继续进入售票厅的人数**

　　buyer =0，表示⑤  **售票厅中已有30名顾客(购票者)**

　　buyer <0，|buyer|的（绝对）值表示⑥  **等待进入售票厅的人数**

若欲购票者最多为n个人，则信号量buyer的最大值为⑦  **30**

信号量buyer的最小值为⑧ **30-n**

1. 有一个理发店 , 店内共有 N个座位供顾客等待理发, (进入理发店的顾客 , 都在此座位上等待理发, 正在理发的顾客不占用此座位) ,当 N个座位坐满了, 后到的顾客不等待,立即回家。当没有顾客时,理发师睡眠等待。

顾客信号量 = 0

理发师信号量 = 0

互斥信号量mutex = 1 // 椅子是理发师和顾客精进程都可以访问的临界区

int 空椅子数量 = N //所有的椅子数量

理发师（线程/进程）

While(true){ //持续不断地循环

P（顾客） //试图为一位顾客服务，如果没有他就睡觉（进程阻塞）

P（互斥信号量） //如果有顾客，这时他被叫醒（理发师进程被唤醒），要修改空椅子的数量

空椅子数量++ //一张椅子空了出来

V（理发师） //现在有一个醒着的理发师，理发师准备理发，多个顾客可以竞争理发师互斥量，但是只有一个顾客进程可以被唤醒并得到服务

V（互斥信号量） //释放椅子互斥量，使得进店的顾客可以访问椅子的数量以决定是否进店等待

/\* 理发师在理发 \*/

｝

顾客（线程/进程）

while(true)

｛ //持续不断地循环

P（互斥信号量） //想坐到一张椅子上

if (空椅子数量 > 0)

{ //如果还有空着的椅子的话

空椅子数量-- //顾客坐到一张椅子上了

V（顾客） //通知理发师，有一位顾客来了

V（互斥信号量） //顾客已经坐在椅子上等待了，访问椅子结束，释放互斥量

P（理发师） //该这位顾客理发了，如果理发师还在忙，那么他就等着（顾客进程阻塞）

/\* 竞争到了理发师则该顾客开始理发 \*/

}

else

｛ //没有空着的椅子

V（互斥信号标） //不要忘记释放被锁定的椅子

/\* 顾客没有理发就走了 \*/

｝

｝