Mohamed Boudiaf University – M'sila
Department of Computer Science
3rd year, Computer Science - SI

SOLUTION	**TEST I / G 1**

We consider a system with three parallel processes **P1, P2, and P3**, such that:

- **P1 and P2 cannot read the File before P3 has written to it.**
- **P2 and P3 manipulate the same variable "x".**
- **P1 prints the message after 2 iterations of P2**
- **P1 consumes "object" only after three productions of it by P3.**

The goal is to solve the synchronization problem using **at most 5 semaphores**. Note that the production **buffer is unlimited**, so we can produce and consume without worrying about its size.

Var

s1, s2, s3, s4 : semaphore := 0 ; mutex : semaphore := 1 ;

Process P1;	Process P2;	Process P3;
Begin	Begin	Begin
P(s1);	**P(s2);**	write (File);
read (File);	read (File);	**V(s1); V(s2);**
P(s3); P(s3);	**P(mutex);**	**P(mutex);**
print ("2 iterations of P2");	x++;	print (x);
P(s4); P(s4); P(s4);	**V(mutex);**	**V(mutex);**
Consume(object);	**V(s3);**	Produce (object);
End;	End;	**V(s4);**
		End;

Mohamed Boudiaf University – M'sila
Department of Computer Science
3rd year, Computer Science - SI

SOLUTION	**TEST I / G 2**

We consider a system with three parallel processes **P1, P2, and P3**, such that:

- **P3 cannot execute the receive task before P1 and P2 have sent their packages.**
- **P1 and P2 write to the same file.**
- **P3 prints the message "triple writes" after 2 writes by P2 and one write by P1.**
- **P3 prints the message "done" only after P1 has sent its package and P3 have received it.**

The goal is to solve the synchronization problem using **at most 5 semaphores**. Note that the **send-receive buffer is unlimited**, so we can send and receive without worrying about its size.

Var

s1, s2, s3, s4 : semaphore := 0 ; mutex : semaphore := 1 ;

Process P1;	Process P2;	Process P3;
Begin	Begin	Begin
Repeat	Repeat	Repeat
send (package1);	send (package2);	**P(s1); P(s2);**
V(s1);	**V(s2);**	Receive (package1, package2);
P(mutex);	**P(mutex);**	print ("done");
write (File);	write (File);	**P(s3); P(s3); P(s4);**
V(mutex);	**V(mutex);**	print ("triple writes");
V(s4);	**V(s3);**	Until false;
Until false;	Until false;	End;
End;	End;	

Mohamed Boudiaf University – M'sila
Department of Computer Science
3rd year, Computer Science - SI

SOLUTION	**TEST I / G 3**

We consider a system with three parallel processes **P1, P2, and P3**, such that:

- **P1 produces "object1", then P3 consumes it.**
- **P1, P2 and P3 manipulate the same counter "count"**
- **P3 consumes "object2" only after two productions of it by P2.**
- **P3 prints the counter "count" after P1 has incremented it.**

The goal is to solve the synchronization problem using **at most 4 semaphores**. Note that the production **buffer is unlimited**, so we can produce and consume without worrying about its size.

Var

s1, S2, s3 : semaphore := 0 ;

mutex : semaphore := 1 ;

Process P1;	Process P2;	Process P3;
Begin	Begin	Begin
repeat	Repeat	Repeat
produce(object1);	**P(mutex);**	**P(s1);**
V(s1);	Count--;	Consume(object1);
P(mutex);	**V(mutex);**	**P(s2); P(s2);**
Count++;	Produce(object2);	Consume(object2);
V(mutex);	**V(s2);**	**P(s3);**
V(s3);		**P(mutex);**
Until false;	Until false;	Print(Count);
End;	End;	**V(mutex);**
		Until false;
		End;