1. Considere un sistema con intercambio en el cual la memoria consiste en particiones (huecos) con las siguientes longitudes: 6K, 7K, 2K, 18K, 30K. Estos huecos se encuentran en el orden indicado. Diga qué huecos son tomados para las siguientes solicitudes: a) 2K b) 10K c)17K. Utilice los algoritmos de primer acceso (**FF**), mejor acceso (**BF**), peor acceso (**WF**).

2. Si un sistema con intercambio tiene huecos libres con las siguientes longitudes: 10K, 4K, 20K, 18K, 7K, 9K, 12K, 15K en ese orden. Que particiones son tomadas para las siguientes solicitudes: a) 12K b) 10K c) 9K considerando los algoritmos de primer acceso (**FF**), mejor acceso (**BF**), peor acceso (**WF**).

3. Considerar un sistema con intercambio, en el que la memoria posee particiones libres de tamaño fijo: 1000Kb, 400Kb, 1800Kb, 700Kb, 900Kb, 1200Kb y 1500Kb. Estos huecos están dispuestos en el orden dado. Se tienen tres procesos de tamaños 1200Kb, 1000Kb y 900Kb. Para los algoritmos:
   - Primero en ajustarse
   - Mejor en ajustarse
   - Peor en ajustarse
   - Siguiente en ajustarse

   a) ¿Qué huecos serán asignados?
   b) ¿Qué algoritmo aprovecha mejor la memoria?

4. Dado un sistema con:
   Dirección virtual: 32 bits .
   Tamaño de página = 4 Kbytes (2^{12} bytes).
   Determinar:
   a) Bits necesarios para el desplazamiento
   b) Bits necesarios para el número de páginas.
   c) Número posible de páginas virtuales

5. Dado un sistema con la siguiente tabla de páginas, obtener las direcciones fÌsicas correspondientes. Las direcciones virtuales tienen 4 bits de página y 12 bits de ajuste.

Página M. Virtual	Página M. Física	Bit presencia
15	000	0
14	000	0
13	000	0
12	000	0
11	111	1
10	000	0
9	101	1
8	000	0
7	000	0
6	000	0
5	011	1
4	100	1
3	000	1
2	110	1
1	001	1
0	010	1

Direcciones Virtuales:
a) 400h
b) 0BF4h
c) 1101010011100000
d) 1011001110010010
e) 20895
f) 0B120h

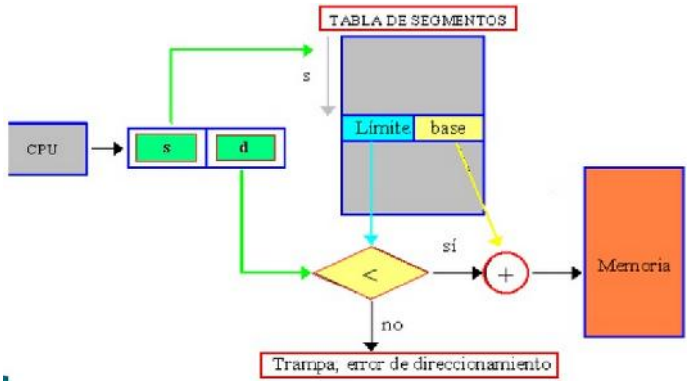
6. (Paginación) Supóngase que la tabla de páginas de un proceso que se estáejecutando en el procesador es:

Nº de Página	Nº de Marco	R	M	V
0	4	1	0	1
1	7	1	1	1
2	---	0	0	0
3	2	0	0	1
4	---	0	0	0
5	0	0	1	1

R: es el bit de referencia. R=1 (la página ha sido referenciada)
M: es el bit de modificación. M=1 (la página ha sido modificada)
V: es el bit de Presente/ausente. V=1 (la página en cuestión está en memoria principal, tiene un marco asociado)
El tamaño de las páginas es de 1024 bytes. El marco 0 está cargado el la dirección física cero y el resto sucesivamente
Se pide: ¿A qué direcciones físicas corresponden las siguientes direcciones virtuales?
a). (1, 125) b). (2, 324) c). (5, 322) d). (7, 321) e). (3, 1026) El formato en el que se da la dirección virtual corresponde a (nº de página, desplazamiento)

7. (Segmentación) Considerando la siguiente tabla de segmentos, determine las direcciones físicas:

Segmento	Límite	Base
0	600	219
1	14	2300
2	100	90
3	580	1327
4	96	1952



a) (0,430)
b) (1,10)
c) (3,400)
d) (2,500)
e) (4,112)

El formato de la dirección es (segmento, desplazamiento)

8. (Segmentación y paginación) Se tiene un sistema operativo multitarea con las siguientes características:
- Gestión de memoria virtual formada por la combinación de la segmentación y la paginación.
- Direcciones virtuales de 32 bits (igual que las direcciones físicas) con el siguiente formato:

8 bits	12 bits	12 bits
Nº de segmento	Nº de página	Desplazamiento

a) ¿Cuántos segmentos distintos puede direccionar cualquier proceso del sistema?
b) ¿Cuántos marcos de página distintos puede haber como máximo?

3.

Para el algoritmo primero en ajustarse:

Segmento	Hueco asignado	Fragmentación
1200Kb	1800	600
1000Kb	1000	0
900Kb	900	0

Para el algoritmo mejor ajustarse:

Segmento	Hueco asignado	Fragmentación
1200Kb	1200	0
1000Kb	1000	0
900Kb	900	0

Para el algoritmo peor en ajustarse:

Segmento	Hueco asignado	Fragmentación
1200Kb	1800	600
1000Kb	1500	500
900Kb	1200	300

Para el algoritmo siguiente en ajustarse:

Segmento	Hueco asignado	Fragmentación
1200Kb	1800	600
1000Kb	1200	200
900Kb	1500	600

Si el sistema de gestión de memoria es de particiones fijas (como indica el enunciado) el problema es la fragmentación interna. En este caso el algoritmo que mejor aprovecha la memoria es el mejor en ajustarse dado que la fragmentación interna es nula, después el algoritmo primero en ajustarse con fragmentación interna de 600 y finalmente los otros dos que producen una fragmentación interna de 1400

4. Dirección virtual: 32 bits
   a) si la página es de 4kb es necesario 12 bits para el desplazamiento
   b) 32bits – 12bits = 20 bits para el número de páginas
   c) 2^{20}= 1048576

5. a) dirección física 400h= 0000 0100 0000 0000
   Dirección virtual = 0010 0100 0000 0000 = 2400h

   b) dirección física 0BF4h= 0000 1011 1111 0100
   Dirección virtual = 0010 1011 1111 0100 = 2BF4h

   c) dirección física = 1101 0100 1110 0000 = 0D4E0h
   Dirección virtual = error. No está presente la página

   d) dirección física = 1011 0011 1001 0010 = 0B392h
   Dirección virtual = 0111 0011 1001 0010 = 7392h

   e) dirección física 20895 = 519Fh = 0101 0001 1001 0101
   Dirección virtual = 0011 0001 1001 0101 = 319Fh

   f) dirección física 0B120h= 1011 0001 0010 0000
   Dirección virtual = 0010 0001 0010 0000 = 2120h

6. Paginación
   a). (1, 125) = 7*1024+125= 7293
   b). (2, 324) = fallo de página. La página 2 no tiene ningún marco asociado,

c). (5, 322) = 0*1024+322= 322
d). (7, 321) = error de direccionamiento. No existe la página 7.
e). (3, 1026) = error de direccionamiento. El desplazamiento 1026>1024 por lo que es mayor que el tamaño de marco

7. Segmentación
a) (0,430) = 219+430= 649
b) (1,10) = 2300+10= 2310
c) (3,400) = 1327+400= 1727
d) (2,500)= error de direccionamiento 500<100
e) (4,112) = error de direccionamiento 112<96

8. Segmentación paginada

32bits:	8 bits	12 bits	12 bits
	Nº de segmento	Nº de página	Desplazamiento

a) ¿Cuántos segmentos distintos puede direccionar cualquier proceso del sistema?
b) ¿Cuántos marcos de página distintos puede haber como máximo?
a) $2^8 = 256$
b) Un razonamiento puede ser:
como cada segmento direcciona a una tabla de páginas:
total de marcos = $2^8 * 2^{12} = 2^{20} = 1048576$
otra forma es calcular la cantidad total de memoria / el tamaño de cada página:
total de marcos = $2^{32} / 2^{12} = 2^{20} = 1048579$

1- **En** un esquema de segmentación paginada con páginas de 1Kb, ¿Es posible que la dirección lógica (2, 1333) se pudiera traducir a la dirección física 3654? ¿Y a la dirección física 2357?. Razonarlo

2- **S**ea un sistema de memoria segmentado-paginado. Los espacios de direcciones lógicas son de 8Gbytes y el tamaño de página es de 2Kbytes. Un espacio de direcciones puede tener hasta 256 segmentos y los descriptores de página tienen 4 bytes. Diga cuál será el tamaño de la tabla de páginas de un segmento.

3- Una computadora provee a cada proceso con un espacio de direccionamiento de 65536 bytes dividido en páginas de 4096 bytes. Un programa particular tiene un código de 32768 bytes, datos 16386 bytes y pila de 15870 bytes. ¿Entrará este programa en el espacio de direcciones? Si el tamaño de cada página fuera de 512 bytes, ¿entraría? Recordar que una página no puede contener partes de dos segmentos diferentes.

4- Describa cuál es la forma de detectar una dirección no válida en los siguientes métodos de gestión de memoria.
   a) Paginación.
   b) Segmentación.
   c) Segmentación paginada.

Respuestas

1. No es posible 3654, si es posible 2357. Para que las direcciones físicas 3654 y 2357 sean accedidas como respuesta a la emisión de la dirección lógica (2, 1333) en este sistema, los desplazamiento dentro del marco y de la página han de coincidir.

Dirección lógica	Campos de la dirección lógica	Dirección física (*)
(2, 1333)	Segmento =2 Página = 1 = 1333 div 1024 desplaz. = 309 = 1333 mod 1024	3654 marco =3 =3654 div 1024 desplaz.= 582= 3654 mod 1024
		2357 marco =2=2357 div 1024 desplaz.=309 = 2357 mod 1024

(*) Para que las direcciones físicas sean válidas sus desplazamiento deben ser de 309, es decir exactamente igual al de la lógica.

2. La solución es 64Kb.
   Razonamiento: Será necesario conocer el número máximo de páginas que puede tener un segmento y multiplicarlo por el tamaño del descriptor de segmento. Para conocer el número máximo de páginas será necesario conocer el número de bits destinado a la página en la dirección lógica.
   Como el espacio de direcciones lógicas es de 8Gb -> 33 bits.
   Como el número máximo de segmentos de un proceso 256 segmentos -> 8 bits de la dirección para el número de segmento.
   Si el tamaño de página es de 2Kb ->11 bits de la dirección para el desplazamiento dentro de la página.
   El número de bits para el nº de páginas de la tabla vendrá dado por: 33 – (8 + 11) = 33 – 19 = 14
   Con lo que los campos y bits de la dirección lógica quedarían:

Segmentos (8bits)	Páginas (14bits)	Desplazamiento (11bits)

   Por lo tanto se tiene un total de 14 bits para la página y se necesitarán 2^{14} = 16384 descriptores de páginas por segmento. Como cada descriptor de página tenía 4 bytes se tiene que ll tamaño de la tabla de páginas = 16384 * 4 bytes = 65536 = **64 KB**

3. Con páginas de 4096 bytes (En total el sistema tendría 16 páginas: 65536 / 4096):
   · Para texto 32768 / 4096 = 8 páginas.
   · Para datos 16386 / 4096 = 5 páginas.
   · Para pila 15870 / 4096 = 4 páginas.
   El total de páginas es de 17, por lo que **no se puede** porque tengo 16

   Con páginas de 512 bytes (En total el sistema tendría 128 páginas: 65536 / 512):
   · Para texto 32768 / 512 = 64 páginas.
   · Para datos 16386 / 512 = 33 páginas.
   · Para pila 15870 / 512 = 31 páginas.
   Por lo tanto preciso en total 128 páginas y **entraría** porque 128 necesarias <= 128 páginas disponibles

4. a) En paginación la forma de detectar una dirección no válida es mediante la comprobación en su tabla de páginas de la no existencia de la página a la que pertenece dicha dirección, para el proceso que la ha emitido.
   b) En segmentación la forma de detectar una dirección no válida es mediante la comprobación de que el desplazamiento de la dirección lógica emitida ha de ser menor que el tamaño del segmento correspondiente, el cual se encuentra almacenado en su tabla de segmentos.
   c) En segmentación paginada la forma de detectar una dirección no válida es la misma que se realiza en segmentación (apartado b).

Actividades

1- Una computadora tiene 4 marcos de página. Para cada uno se muestra el tiempo en que se cargó, el tiempo en que se realizó el último acceso y los Bit R y M (los tiempos se dan en clocks de reloj).

Página	Cargado	Referenciado	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

2 - ¿Qué página será reemplazada con los algoritmos NRU, FIFO, LRU y segunda oportunidad?

3 - Si el algoritmo de reemplazamiento de páginas FIFO es usado con 4 marcos de páginas.
¿Cuántos fallos de páginas ocurrirán con la lista de referencias 0-1-7-2-3-2-7-1-0-3 si los marcos
están inicialmente vacíos? Repita el problema con el algoritmo LRU.

4 - Ídem al anterior utilizando 5 marcos de página.

5 - Determine cuántos fallos de página se producirán con la siguiente lista de referencias
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1 considerando los algoritmos FIFO, óptimo y LRU.
Suponga primero que dispone de 3 marcos de página y repítalo para 4 marcos de páginas.

6 - Ídem al anterior para la siguiente lista 0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 2 3 4 1.

Página	Cargado	Referenciado	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

- ¿Qué página será reemplazada con los algoritmos **NRU**, **FIFO**, **LRU** y **segunda oportunidad**?

a) ¿Cuál pagina se remplazará si se usa NRU?

Este algoritmo favorece a las páginas que fueron usadas recientemente. Funciona de la siguiente manera: cuando una página es referenciada, fija el bit de referencia para esa página. Similarmente, cuando una página es modificada, fija su bit de modificación. Usualmente estas operaciones son realizadas por el hardware, aunque puede hacerse también por software. En un tiempo fijo, el sistema operativo pone en 0 los bits de referencia de todas las páginas, de modo que las páginas con su bit de referencia en 1 son las que fueron referenciadas dentro del último intervalo de reloj. Cuando una página debe ser reemplazada, el sistema operativo divide las páginas en cuatro categorías:

- Categoría 0: no referenciada, no modificada
- Categoría 1: no referenciada, modificada
- Categoría 2: referenciada, no modificada
- Categoría 3: referenciada, modificada

Las mejores páginas para cambiar son las que se encuentran en la categoría 0, mientras que las peores son las de la categoría 3. Se desaloja al azar una página de la categoría más baja que no esté vacía. Este algoritmo se basa en la suposición de que es mejor desalojar una página modificada a la que no se ha hecho referencia en al menos un tic de reloj, en vez de una página limpia que se está usando mucho.

R= 0

b) ¿Cuál pagina se remplazará si se usa FIFO?

En este método, el sistema operativo sólo tiene que guardar en orden las páginas que fueron cargadas, de modo que al necesitar hacer espacio pueda fácilmente elegir la primera página cargada. Se usa una cola, al cargar una página nueva se ingresa en el último lugar. Aunque las colas FIFO son simples e intuitivas, no se comportan de manera aceptable en la aplicación práctica, por lo que es raro su uso en su forma simple. Uno de los problemas que presentan es la llamada Anomalía FIFO o Anomalía de Belady. Belady encontró ejemplos en los que un sistema con un número de marcos de páginas igual a tres tenía menos fallos de páginas que un sistema con cuatro marcos de páginas. El problema consiste en que podemos quitar de memoria una página de memoria muy usada, sólo porque es la más antigua.

R=2

c) ¿Cuál pagina se remplazará si se usa LRU?

Este algoritmo difiere del de 'No usada recientemente' en el hecho de que aquel sólo se fija en el intervalo de tiempo desde que se pusieron en 0 los bits de referencia de las páginas, mientras que el algoritmo de 'Menos usada recientemente' intenta proveer un comportamiento casi óptimo mediante la observación de las páginas que menos fueron usadas recientemente. Este tipo de páginas, estadísticamente son las que tienen menor probabilidad de ser usadas nuevamente.

R=1

d) ¿Cuál pagina se remplazará si se usa 2da oportunidad?

Es una pequeña modificación al algoritmo FIFO, que funciona bastante mejor que el FIFO. En este caso cuando una página debe ser sacada se toma la primera en la cola, y en vez de sacarla, consulta el valor de un bit de referencia. En caso de estar fijado (en 1) se cambia el bit a 0 y se lo coloca al final de la obstrucción, actualizando su tiempo de carga como si recién hubiera llegado al procesador. De esta forma, se le da una segunda oportunidad. Si el bit se encuentra sin fijar(en 0), la página se saca de memoria. Cada vez que la MMU accede a una página, fija su bit de referencia a 1. Para esto es necesario soporte para bit de referencia por hardware.

R=0

2)

FIFO

	0	7	2	3	2	7	1	0	3	
M1	0	7	2	3	3	3	1	0	0	
M2		0	7	2	2	2	3	1	1	
M3			0	7	7	7	2	3	3	
M4				0	0	0	7	2	2	
	F	F	F	F	-	-	F	F	-	6F

LRU

	0	7	2	3	2	7	1	0	3	
M1	0	0	0	0	0	0	1	1	1	
M2		7	7	7	7	7	7	7	7	
M3			2	2	2	2	2	2	3	
M4				3	3	3	3	0	0	
	F	F	F	F	-	-	F	F	F	7F

OPTIMO

	0	7	2	3	2	7	1	0	3	
M1	0	0	0	0	0	0	0	0	0	
M2		7	7	7	7	7	1	1	1	
M3			2	2	2	2	2	2	2	
M4				3	3	3	3	3	3	
	F	F	F	F	-	-	F	-	-	5F

a) Indicar la salida por pantalla y el valor final de los semáforos.
Suponer que inicialmente todos los semáforos tienen valor cero.

Proceso 1	**Proceso 2**	**Proceso 3**
printf("A");	sem_wait(&s1);	sem_wait(&s2);
sem_post(&s3);	printf("D");	sem_wait(&s4);
printf("B");	sem_wait(&s3);	printf("E");
sem_post(&s2);	sem_post(&s4);	printf("C");
sem_post(&s1);	sem_wait(&s3);	sem_post(&s3);

b) Considera un sistema de paginación en el que las direcciones lógicas son de 22 bits y el tamaño de página es de 2 Kbytes. Sabiendo que cada byte se direcciona independientemente, calcula el ahorro de memoria que obtendríamos para representar la tabla de páginas de un proceso que está utilizando 90 Kbytes de memoria, cuando empleamos una tabla de páginas con dos niveles en lugar de tener una tabla de un solo nivel. En el sistema con dos niveles, debes considerar que se emplean 5 bits de la dirección para el segundo nivel. Además, cada entrada de las tablas de páginas precisa 8 bytes. Razona la respuesta.

c) Considera un sistema de paginación en el que las direcciones lógicas son de 20 bits y el tamaño de página de 4 Kbytes. Sabiendo que cada byte se direcciona independientemente, calcula el ahorro de memoria que obtendríamos para representar la tabla de páginas de un proceso que está utilizando 192 Kbytes de memoria, cuando empleamos un tabla de páginas con dos niveles en lugar de tener una tabla de un solo nivel. En el sistema con dos niveles debes considerar que se emplea el mismo número de bits de la dirección para cada nivel. Cada entrada de las tablas de páginas precisa 16 bytes. Razona la respuesta.

Respuestas:

a) Salida: ABDEC y todos los semáforos quedan en 0

b) Si el tamaño de página son 2KB, esto implica que de los 22 bits de la dirección lógica, 11 bits se utilizan para direccionar la página, por lo que quedan 11 bits para codificar la tabla de páginas.
Si se utiliza una tabla de páginas de un sólo nivel, su tamaño es el número de entradas de la tabla por el tamaño de la entrada que era de **8** bytes, es decir,
$2^{11} * 8 = 2^{14}$ bytes. = 16384 bytes
Si se utiliza una tabla de páginas de dos niveles, 6 bits de la dirección lógica se emplean para el primer nivel ya que el enunciado dice que 5 bits son para el segundo nivel. Entonces, cada tabla de segundo nivel direcciona hasta 2^5 páginas, es decir 32 páginas de 2KB cada una (total 64KB). Como el proceso requiere 90KB, hacen falta 2 tablas de segundo nivel. Así, el consumo es el de **una** tabla de primer nivel más el de **dos** de segundo nivel, esto es,
$2^{6} * 8 + 2 * 2^5 * 8 = 2^{10}$ bytes = 1024 bytes
Por lo tanto, el ahorro al utilizar una tabla de páginas de dos niveles es de $2^{14} - 2^{10}$ bytes = 16384 -1024 =. **15360 bytes**

c) Como el tamaño de la pagina es de 4 KB, de los 20 bits de dirección lógica, 12 bits son para direccionar el contenido de la pagina (2^{12}=4Kb) y, por lo tanto, 8 bits son para direccionar las páginas.
Si el sistema tiene una tabla de un solo nivel, la tabla tiene 256 entradas, y su coste de almacenamiento es de **$256 * 16 = 2^{12}$ bytes = 4096 bytes**. (recordar que cada entrada ocupaba 16bytes)
Si el sistema tiene una tabla de dos niveles, la tabla de primer nivel tiene 2^4 entradas es decir 16 entradas, al igual que la de segundo nivel. Si un proceso utiliza 192 KB, se obtiene que necesita 192KB / 4KB = 48 páginas, y se emplean 3 tablas de segundo nivel. Ahora el consumo es de **una** tabla de primer nivel y 3 de segundo, es decir,
$2^4 * 16 + 3 * 2^4 * 16 = 2^{10}$ bytes = 1024 bytes
Por lo tanto, el ahorro es $2^{12} - 2^{10}$ bytes = 4096 - 1024 = **3072 bytes**

1- Una cochera subterránea inteligente cuenta con capacidad para 10 autos, y un montacargas que sube y baja a los autos. Dicho montacargas solo puede ser utilizado por un auto a la vez, ya sea para subir o para bajar. Determine los semáforos necesarios con sus valores de inicio para sincronizar dicha cochera.
Utilice las funciones: salir_de_la_cochera() y entrar_a_la_cochera()

2- Escribir el algoritmo “productor consumidor” usando semáforos.

3- En un bar el vendedor atiende pedidos, donde los clientes van de uno en uno. La operatoria es la siguiente: cuando llega un cliente, debe esperar a que el vendedor esté libre para que le pueda prestar atención al pedido. Cuando hace el pedido el vendedor busca el producto y le informa el precio. El cliente espera a que le informe el precio y busca el dinero para pagar. Mientras tanto el vendedor esperará a que el alumno abone el precio del producto. Y cuando el cliente recibe el producto deberá esperar lugar para sentarse en una mesa si es que no hay para poder consumir su pedido. Luego podrá marcharse del bar.

1- Solución

Semáforos	Valor inicial
cochera	10 (suponiendo que la cochera está vacía)
vehiculo	0 (suponiendo que la cochera está vacía)
montacargas	1 (mutex)

```
Proceso entrar()
        wait(cochera)
        wait(montacargas)
        entrar_a_la_cochera()
        post(montacargas)
        post(vehiculo)
fin proceso
```

```
Proceso salir()
        wait(vehiculoo)
        wait(montacargas)
        salir_de_la_cochera()
        post(montacargas)
        post(cochera)
fin proceso
```

2- Solución:

Semáforos:
lleno = 0;
vacio = N;
mutex = 1;

```
Productor( ) {
    While (1) {
        x = producir( );
        P(vacio);
        P(mutex);
        Ingresar (x);
        V(mutex);
        V(lleno);
    }
}
```

```
    Consumidor( ){
        While (1){
            P(lleno);
            P(mutex);
            x= sacar ( );
            V( mutex);
            V(vacio);
            consumir (x);
        }
    }
```

3-

```
#include <iostream>
#include <pthread.h>
#include <semaphore.h>
#include <cstdio>
#include <cstdlib>
#include <unistd.h>
#include <ctime>
using namespace std;
using namespace std ;

sem_t atencionVendedor;
sem_t asiento;
sem_t pagar;
sem_t precio;
sem_t producto;
sem_t entrega;
int i=1;

void * vendedor( void *) {
   while(1) {
        sem_wait(&producto);
        cout<<"Vendedor busca producto para el cliente "<<i<<endl;
        cout<<"Vendedor busca precio para el cliente "<<i<<endl;
        sem_post(&precio);
        sem_wait(&pagar);
        cout<<"Vendedor recibe e dinero cliente "<<i<<endl;
        sem_post(&entrega);
        sem_post(&asiento);
   }
  return NULL ;
}

void * consumidor( void * ) {
  while(1) {
        sem_wait(&atencionVendedor);
        cout<<"Cliente "<<i<<" elige el producto\n";
        sem_post(&producto);
        sem_wait(&precio);
        cout<<"Cliente "<<i<<" busca el dinero\n";
        sem_post(&pagar);
        sem_wait(&entrega);
        cout<<"Cliente "<<i<<" toma el producto\n";
        cout<<"Cliente "<<i<<" se sienta\n";
        sem_wait(&asiento);
        cout<<"Cliente "<<i<<" consume el producto\n";
        cout<<"cliente "<<i<<" sale\n\n";
```

```
            i++;
            Sleep(2000*(rand()%3));
            sem_post(&atencionVendedor);
        }
    return NULL ;
    }

    int main(int argc, char **argv) {
            srand(time(0));
            sem_init(&atencionVendedor,0,1);
            sem_init(&asiento,0,0);
            sem_init(&pagar,0,0);
            sem_init(&precio,0,0);
            sem_init(&producto,0,0);
            sem_init(&entrega,0,0);
            pthread_t hebras[2];
            pthread_create(&(hebras[0]), NULL, vendedor, NULL);
            pthread_create(&(hebras[1]), NULL, consumidor, NULL);
            for(unsigned i=0; i < 2; ++i)
               pthread_join(hebras[i], NULL);
            sem_destroy(&atencionVendedor);
            sem_destroy(&asiento);
            sem_destroy(&pagar);
            sem_destroy(&precio);
            sem_destroy(&producto);
            sem_destroy(&entrega);
            return 0;
    }
```