PART 1 - Programa concurrente:

-Proceso: programa secuencial que forma parte del programa concurrente. 166831
-Programa: conjunto de procesos dada la definición anterior. 166831
-Paralelismo: Un procesador ejecuta una tarea y otro procesador ejecuta otra simultáneamente. **168158**

-Concurrencia: Tareas que se realizan en orden de dependencia (la 2a depende la 1a). Puede ser paralelismo. 167319

-Overlapping: Tareas que son independientes pero comparten recursos. 167319

-Multiprocesadores: Sistemas compuestos por múltiples procesadores que se sincronizan y distribuyen el trabajo. 166912

-Multitasking: Generalización simple del concepto de superponer inputs y outputs (I/O) mediante cómputo para superponer la ejecución de un programa con la de otro. 168663

-Procesador Gráfico: Podriamos definirlo como una computadora que su especialidad y tarea es tomar información de la memoria de la computadora y mostrarla en la pantalla.168724

-Ejemplos de Multiprocesadores: En primer ligar un ejemplo claro es en aplicaciones cientificas como lo san las que se ultilizan para el pronostico del clima y estudio de este .Por otra parte otro ejemplo importante es el "Internet" en su totalidad ya que este trabaja para proporcionar y distruibuir la informacion pero a través de paginas web. 168724

-Multiprocesadores: (nuevo) Sistemas compuestos por múltiples procesadores que se sincronizan y distribuyen el trabajo. 166912

-Programa Ordinario: Consiste en la decalaración de datos, sentencias de asignación y flujo de control en un lenguaje de programación. 163221

- Retos de la programación concurrente: Nnecesidad de sincronizar la ejecución de diferentes procesos y permitirles comunicarse. 163221

- Tipos de procesadores: El procesador gráfico, ya que es una computadora especializada para la tarea de tomar
información de la memoria de la computadora y mostrarla en la pantalla. Existen varias empresas que se dedican en realizar este tipo de procesadores, entre ellos una de las mas famosas es nvidia. Tambien las interfaces interactivas tienen sus propios
procesadores especializados. 170716

- Como funciona una super computadora: Una super computadora funciona a base de multiprocesadores que son sistemas diseñados para llevar la potencia informática de varios procesadores para trabajar en equipo en un solo problema intensivo. 170716

- **Bus del ordenador**: es un dispositivo de transmisión de datos compartido entre varios componentes de un sistema digital, como el procesador, la memoria RAM, los puertos, etc. 403715

- **Un thread:** es proceso ligero o subproceso es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo. 403715

- **Recurso compatido :** entidad (valor, estado...) compartida por varios threads. Por lo general, sólo lo utiliza uno de ellos a la vez. 403783

- **Sincronizacíon :** la idea de coordinar threads para realizar una tarea. En general, un thread estará esperando una acción de otro thread para completar su tarea. 403783

- **Scheduler** : el planificador es el componente del núcleo del sistema operativo que elige el orden de ejecución de los procesos en los procesadores de un ordenador. 403719

- **Typo de scheduler** : Existen principalmente seis tipos de algoritmos de programación de procesos: First Come First Serve (FCFS), Shortest-Job-First (SJF) , Shortest Remaining Time, Priority Scheduling,Round Robin Scheduling,Multilevel Queue Scheduling

Part 2 – Programacion sequencial

Propiedades de programas:
-Safety: Propiedad que se asegura que un programa nunca entra a un mal estado (nada malo pasa) -166831

-Liveness: Propiedad que se asegura que un programa eventualmente llegará a un estado deseable, usualmente de término (algo bueno va a pasar) -166831

-Sección crítica: es una sección protegida en la cual no puede ingresar mas de un proceso a la vez. **168158**

-*Total correctness*: es una propiedad que combina *parcial correctness* (*Safety*) y terminación (*Liveness*) **168158**

Ejemplo **Ausencia de deadlock:** Imagine que tenemos 2 procesos A y B. El proceso A necesita el recurso 1 y el proceso B el recurso 2.
Para continuar su ejecución el proceso A necesita del recurso 2 y el proceso B el recurso 1.
Así que tanto A como B esperan a que se libere el respectivo recurso que necesitan.
Pero como A no ha terminado su proceso no "soltara" el recurso 1, lo mismo sucede con B.
Por lo que ambos procesos se quedaran esperando indefinidamente. **168158**

-Partial correctness: es una propiedad safety , establece que si el programa termina entonces llegamos al resultado correcto. Por otra parte, si el programa falla en terminar es que nunca llegara al resultado correcto **. 165974**

-Termination: es una propiedad liveness, que asegura que el programa es finito, es decir que tiene un fin. **165974**

-Mutual Exclusion: es una propiedad safety, garantiza que no entra más de un proceso a la vez dentro de la sección critica. **165974**

-Axioma: son una secuencia de formulas que se asumen como verdad desde el inicio por lo que no requieren demostración. **165974**

-Predicado lógico: Extiende una proposición lógica para poder manipular expresiones boleanas. 167319

-Programming Logic (Lógica de Programación): Sistema formal que soporta un acercamiento por medio de aserciones para desarrollar y analizar programas.
Asimismo, una Lógica de Programación es un sistema lógico formal que facilita hacer declaraciones precisas sobre la ejecución de un programa **168663**

Elementos de la Programación Secuencial:
-Declarations: sirven para definir los tipos, constantes y variables. 166912

-Statements: Asignan valores a las variables y se encargan de la ejecución del programa (funciones-C, métodos-Java). 166912

-Procedures: Definen parámetros y funciones. 166912

-**Programa en lenguaje c:** Un programa en lenguaje c es aquel que esta formado por una o varias funciones. Lo que se busca con cada funcion es expresar como se realizo el algoritmo que resuelve una parte en las que se ha descompuesto el problema. Un ejemplo de esto es el metodo de analisis descedente, este consiste en resolver un problema que va de lo mas complejo a lo mas simple, hasta que su programacion resulta ser trivial. Y usando lenguaje c cada una de estas parte se resuelve con una funcion. 168724

-**Funcion en c:** Una funcion en c podemos decir que tiene un nombre y se constituye a partir de un conjunto de instrucciones al ejecutar o invocar a estas, obtenemos un valor.Una funcion importante de recordar es la funcion "main", a esta la invocamos desde el sistema operativo, cuando comenzamos con la ejecucion del programa. 168724.

-Variable en c: Una varaible es un espacio de memoria, que es importante declarar antes de usarse para determinar: su espacio de memoria , los valores que puede tener, las operaciones que esta puede hacer y al mismo tiempo como va a realizar estas mismas operaciones a todo esto le podemos agregar que esta tenga un valor inicial. 168724.

-Constantes en C: entidades cuyo valor jamás es modificado durante la ejecución de un prgrama. Estas pueden ser de tipo numérico (1, 1000, -38.4), tipo caracteres ("a", "T", "\n") o incluso de tipo cadena ("hello world"). 163422

-Función sizeof() en C: esta función recibe como parámetro una variable y regresa el número de bytes que ocupa en la memoria esa variable. Resulta particularmente útil por ejemplo cuando se quiere conocer la longitud de un array ya que simplemente se llama a la función y se divide entre el tipo de elemento que esté en el array y de esa manera se encuentra el tamaño total del arreglo. 163422

- método void en java: Es un procedimiento que ejecuta las instrucciones que se le dan, se le especifica qué parámetros maneja (puedes darle algún número entero, algun string, etc) y dentro del void lo puede llegar a modificar o utilizar su información. Lo que la caracteriza es que no regresa un elemento de salida al terminar. 168406

- método int o string (return) en java: Es un procedimiento al cual se le dan parámetros (podrían ser cero) y dentro del cuerpo ejecuta ciertas instrucciones. Lo que lo caracteriza es que al final regresa un nuevo elemento del tipo que se le indica al inicio (un entero, una cadena, etc). 168406

Diagramas de Clases UML

-UML es un lenguaje estándar de modelado que nos permitirá hacer abstracciones del mundo real hacia un "mini mundo" para poder representar sistemas concentrándonos en el diseño y la lógica de este. 163422

- Diagrama de Clases UML

- Tipos de Diagramas: Propósito general (diagramas de clase), propósito especifico (diagramas de tiempo). 163221

-Clase: Las clases son conjuntos de objetos que comparten una estructura y comportamiento comunes, especifican los datos de los objetos junto con sus acciones. En UML una clase se representa por dividida en 3 secciones: la primera contiene el nombre de la clase, la segunda los atributos y la tercera las operaciones o acciones de la clase. 163422

-Atributos: Las clases pueden tener varios o ningún atributo dependiendo de cómo sea necesario representarlas. Los atributos representan de manera general el estado de cualquier objeto. 163422

-Métodos: son las operaciones de la clase que especifican el modo de invocar un comportamiento específico. Por lo general los métodos especifican lo que hará una clase pero no siempre cómo se va a realizar. Es importante notar la diferencia que hace UML entre invocación e implementación de un método. 163422

- **Características de los modelos:** Abstracción de las cosas reales, Ignora detalles relevantes y simplifica el sistema real. 163221

- **agregación :** La agregación permite definir una entidad como vinculada a varias entidades de diferentes clases. Es una generalización de la composición, que no conlleva la pertenencia. 403715

- composición: Una composición es un tipo de enlace que implica una relación en la que el hijo no puede existir independientemente del padre. Ejemplo: Casa (padre) y Habitación (hijo). Las habitaciones no existen separadas de una casa.403715

Elementos de la Programación Secuencial (nuevos):
-Declarations: sirven para definir los tipos, constantes y variables. 166912

-Statements: Asignan valores a las variables y se encargan de la ejecución del programa (funciones-C, métodos-Java). 166912

-Procedures: Definen parámetros y funciones. 166912

Repaso del Lenguaje Java:

Java Technology: Java es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web. Ha sido una opción popular entre los desarrolladores durante más de dos décadas, con millones de aplicaciones Java en uso en la actualidad. Es importante mencionar que java es un entorno de desarrollo, un entorno de aplicación y tambien uno de implementación. Es similar en sintaxis a C++ y se llega a utilizar para desarrollar tanto applets como aplicaciones. 170716

Objetivos de Java: The Java Virtual Machine:
La máquina virtual de Java ejecuta instrucciones generadas por un compilador Java. Consta de un intérprete de bytecode y un entorno de tiempo de ejecución que permiten ejecutar los archivos de clase Java en cualquier plataforma, sea cual sea la plataforma en la que se desarrollaron originariamente. 170716

Garbage Collector: Es el proceso mediante el cual los programas Java realizan la gestión automática de la memoria. Los programas de Java se compilan en un código de bytes que se puede ejecutar en una máquina virtual de Java, o JVM para abreviar. 170716

The Java Runtime Environment: El JRE es la tecnología subyacente que se comunica entre el programa Java y el sistema operativo. Actúa como traductor y facilitador, brindando todos los recursos para que una vez que escriba el software Java, se ejecute en cualquier sistema operativo sin modificaciones adicionales. 17016

PART 3 – Programacion concurrente

Race condition: es una situación en la que el resultado de un programa depende de una ejecución específica - **168158**

Race conditions: Los programas concurrentes son "nondeterministic", los cuales ejecutan ultiples veces el mismo programa concurrente con el mismo numero de inputs, el cual puede crear diversos trazos de ejecución. – **163221**

Date race: ocurre cuando dos hilos acceden a locaciones de memoria compartida. **- 16815**
Trace: Secuencia de estados que muestran la ejecución de un programa concurrente. **- 166831**

No determinismo de programas concurrentes: Los programas concurrentes son no deterministas, es decir, si se ejecuta el mismo programa varias veces, el trace puede ser distinto para distintas ejecuciones. **-166831**

Data races vs race conditions:
- No todos los data races son race conditions y viceversa.
- Race conditions pueden ocurrir incluso cuando no hay acceso a la memoria compartida.
- Data race puede no afectar el resultado. **– 163221**

Semaphores
Semaphore: el semáforo es un tipo de datos abstractos o variables que se utiliza para controlar el acceso a un recurso común por varios subprocesos y evitar problemas de secciones críticas en un sistema concurrente, como un sistema operativo multitarea. **- 170716**

En los semáforos, la justicia "fairness" es opcional. Por esto, podemos encontrar **weak sempahores**(semçaforos debiles) y**strong semaphores** (semáforos fuertes). En los weak, los subprocesos en espera de agotarse no están programadas de manera determinista. En los strong, los subprocesos que esperan para funcionar se programan de manera justa en orden FIFO (primero en entrar, primero en salir). – **166912**

Buscar esto en Permisos en semaforos: Los semáforos ocupan regularmente ciertos permisos que dan acceso a un número finito, esto quiere decir que funciona únicamente con una cierta cantidad de recursos.
Los permisos que otorgan los semáforos cuentan con ciertas propiedades que se deben cumplir:
¨ La capacidad C es el número de recursos disponibles inicialmente
¨ Up (también llamada señal) libera un recurso, que pasa a estar disponible
¨ Down (también llamado esperar) adquiere un recurso si está disponible
-170716

Semaforo Binario- Son semaforos en la que su capacidad C es 1, es decir tienen un solo recurso disponible inicialmente y el contador puede tomar solamente los valores 0 y 1, si toma el valor 0 es que el recurso no esta disponible y si toma el valor 1 entonces el recurso si esta disponible. La diferencia entre un semaforo binario y un candado es que en el semaforo binario cualquier thread puede aumentar el contador a 1 aunque otro thread lo disminuyo a 0, mientras que en el candado solo el thread que esta haciendo uso del candado puede aumentar el contador a 1. Los semaforos binarios garantizan la exclusión mutua y si son de tipo fuerte garantizan que esten libres de starvation. **165974**

Invariantes : El invariante de un objeto podemos definirlo como una propiedad que siempre se cumple entre llamadas a los metodos del objeto.
Dos puntos importantes a destacar de esta propiedad son:

- Cada llamada a un método comienza en un estado que satisface el invariante
- Cada llamada a un método termina en un estado que satisface el invariante

Un ejemplo para esta propiedad seria:
Una cuenta bancaria que no se puede sobregirar tiene un invariante (saldo >= 0) **168724**

Locks:
Candado- es un objeto compartido que le da a los procesos dos operaciones lock y unlock, cuando un proceso entra a la sección critica se usa la operación lock para bloquerla así se garantiza que solo entre un proceso a la vez, ya que termina se utiliza la operación unlock para liberar el candado para que otro proceso pueda utilizarlo. Son llamados mutexes porque garantizan la exclusión mutua. **165974**

Starvation- Es cuando al proceso se le niega el acceso de manera permanente a un recurso que esta solicitando, este concepto se ilustra con la paradoja de la cena de los filosofos de Edsger W. Dijkstra en el caso cuando se adquieren los dos tenedores a la vez, se puede ver que habra algún filosofo que nunca podra adquirir los tenedores . **165974**

El planificador: es un componente funcional muy importante de los sistemas operativos multitarea y multiproceso, y es esencial en los sistemas operativos de tiempo real. Su función consiste en repartir el tiempo disponible de un microprocesador entre todos los procesos que están disponibles para su ejecución, como los threads por ejemplo. 403715

Una pila : es una estructura dinámica de datos LIFO, (una pila), que almacena la información sobre las subrutinas activas de un programa de computadora. Esta clase de pila también es conocido como una pila de ejecución, pila de control, pila de función, o pila de tiempo de ejecución, y a menudo se describe en forma corta como "la pila". Es es un espacio de memoria al que pueden acceder varias entidades. 403715

Monitors:
Un **monitor** proporciona un mecanismo de sincronización estructurado construido sobre construcciones orientadas a objetos.
Proporciona un enfoque estructurado para la programación concurrente, que eleva el nivel de abstracción de la programación
concurrente en comparación con los semáforos. - **166912**

Hay varias disciplinas que se pueden usar en los monitores. Las dos principales son:

- **Señalar y esperar-** donde se garantiza que el la condición señalada se mantiene cuando el subproceso desbloqueado u reanuda la ejecución, porque sigue inmediatamente a la señal.
- **Señalar y continuar-** donde es posible que la condición señalada ya no se mantenga cuando el subproceso desbloqueado u reanuda la ejecución. Esta es la más usada porque es menos compleja. – **166912**

Rule of thumb: Se sugiere evitar nested monitor calls tanto como sea posible. **167319**

Pros y Contras del Uso de monitores.

Pros:

- Los monitores nos brindan un enfoque mas estrucuturado para la programacion concurrente, que se basa principalmente en las nociones de objetos y encapsulacion.
- Con el punto anterior podemos decir que, esto eleva el nivel de abstraccion de la programacion concurrente en comparación con los semáforos.
- La exclusión mutua es algo que esta implicito en el uso de monitores.
- Las variables condicionales un medio claro de sincronizacion

Contras:

- Regularmente los monitores tienen una sobrecarga de rendimiento mayor que los semáforos, es por esto que es importante observar el rendimiento comparado con la propension de error.
- Al tener diferentes disciplinas de señalizacion, esto en ocasiones provoca confusion, esto de alguna manera obstaculiza la abstraccion del monitor.
- Para patrones de sincronizacion complejos, el llamar un monitor anidado es otra fuente de problemas.