Preguntas grupo 5: (Racca, Vargas, Vietto, Zuliani)

Tema: Computation

1) Teniendo en cuenta la estructura de la computadora, esta está formada por:

a) Una memoria central que contiene programas y datos
b) Una unidad central de proceso que ejecuta programas
c) Canales de entrada / salida para los intercambios de información con el exterior
d) Todas las anteriores (opción correcta)

2)¿Cuál es la diferencia entre un lenguaje de programación en alto nivel, bajo nivel y máquina? Mencione algunos ejemplos de cada uno.

La diferencia entre High Level Language y Low Level Language es High Level Language es un lenguaje “amigable” para el programador que proporciona un alto nivel de abstracción del hardware, mientras que Low Level Language es un lenguaje “amigable” para la máquina y no proporciona menos abstracción del hardware. El lenguaje de máquina es un lenguaje propio de cada computadora y se basa en la secuencias de números binarios es decir 0 y 1. Ejemplos de lenguajes bajo nivel C, C++. E.A. Ejemplos de lenguaje de alto nivel Java, C#, Python, JavaScript.

3) ¿Qué unidad se encarga de ejecutar operaciones lógicas simples entre registros de memoria?

La Unidad aritmética y lógica o ALU. Circuito que hace un conjunto de operaciones aritméticas y lógicas con los datos almacenados dentro del procesador.

4)¿Cuál es la diferencia entre un compilador, intérprete y un debugger?

El compilador es un programa que transforma el código fuente de un programa a su equivalente en código de máquina antes de ejecutarlo, entonces, el procesador ejecuta el software obteniendo todas las instrucciones en código de máquina antes de comenzar. El compilador determina qué instrucciones van a enviarse al procesador y en qué orden.

El intérprete es el que ejecuta las instrucciones escritas en el lenguaje de programación dado en tiempos de ejecución, procesa el código línea por línea, de modo que lee, analiza y prepara cada secuencia de forma consecutiva para el procesador.

El debugger revisa el código en busca de errores que impidan que el código funcione de forma adecuada. De esta forma es posible determinar lo que está ocurriendo dentro del código fuente y obtener sugerencias de acciones para mejorarlas. Tiene funciones como los break points, que son puntos en donde el programa se detiene y verifica el estado del programa en ese punto e incluso se puede ver el contenido de una determinada variable.

5)¿ A qué nos referimos cuando hablamos de programación orientada a objetos (POO)?

Defina brevemente los siguientes conceptos: abstracción, encapsulamiento, modularidad, jerarquía, clases y objetos.

La POO es un paradigma de la programación que innova la forma de obtener resultados. Se compone de objetos, donde cada uno de ellos es una entidad que tiene unas propiedades particulares, atributos y métodos, estos objetos manipulan los datos de entrada para la obtención de datos de salida, donde cada objeto ofrece una funcionalidad especial.

Abstracción: Capacidad para centrarse en las características o propiedades esenciales de un objeto, dejando de lado otras no tan relevantes para el dominio del problema. Forma de ver las cosas que están en el problema que estoy queriendo resolver y tiene que ver con la perspectiva de quien lo mira. Herramienta para detectar las clases, es decir, las entidades principales del dominio del problema que tengamos al frente.
Encapsulamiento: Es el mecanismo mediante el cual ocultamos la implementación de un objeto. En programación se logra mediante el Principio de Ocultamiento de Información. Los objetos son cajas negras que nos van a servir para resolver un problema.
Modularidad: Es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos (deben hacer algo pero no más de lo que necesito) y débilmente acoplados (que uno no dependa de los otros). Criterio con el cual voy a modular el problema en entidades que se unan entre sí para resolver el problema.
Jerarquía: Es una clasificación u ordenación de abstracciones. Nuestros programas se construyen como una jerarquía de clases.
Clases: Molde, o definición a partir de la cual se modelan las entidades/objetos. Constituyen nuestro bloque mínimo de código Java. Posee atributos que son las variables que van a describir ese objeto, y métodos.
Objeto: Es una instancia de una clase que tiene: estado (conjunto de todos sus atributos con un valor), comportamiento (métodos o conjunto de mensajes a los que va a poder responder ese objeto) e identidad (dirección de memoria donde va a estar guardado el objeto y lo hace único).

6)¿Cuáles son las ventajas y desventajas del uso de punteros en la estructura de datos?

Optimización: los punteros proporcionan una ventaja de rendimiento por lo que le permite acceder a la memoria de la computadora directamente. En un programa de ordenador, la manera más rápida para acceder y modificar un objeto es para acceder directamente a la memoria física donde se almacena ese objeto. Esta técnica se utiliza comúnmente para optimizar los algoritmos que requieren acceso frecuente o repetitivo a grandes cantidades de datos.
Seguridad: el acceso directo a la memoria significa que puede hacer cosas que tal vez no debería hacer, es decir, dependiendo del lenguaje, el compilador y la plataforma informática, se puede acceder a la memoria sin querer (o intencionalmente). Como resultado, se puede sobrescribir la memoria crítica, modificar el código de una aplicación en ejecución, o hacer que la aplicación u otra aplicación que se comporten de forma inesperada o salida.
Gestión de la memoria: la gestión de uso de memoria en una aplicación que utiliza punteros. Asignar y desasignar la memoria según sea necesario durante el tiempo de ejecución le permite crear objetos de gran tamaño, tales como matrices, de forma rápida e

inmediatamente para liberar la memoria cuando ya no es necesaria. Sin embargo, también es fácil de crear pérdidas de memoria mediante la reasignación de un puntero sin liberar la memoria que estaba apuntando a la primera.
Parámetros de funciones: las funciones pueden devolver un solo valor, pero pueden tener múltiples parámetros. Con el uso de punteros a variables como parámetros, una función puede ser utilizada para establecer los valores de esas variables, y los nuevos valores persistirá después se devuelve la función. Ser capaz de establecer el valor de varias variables a la vez con una sola llamada de función es limpio y eficiente. Sin embargo, puede ser un poco confuso para leer porque no se puede saber si las variables pasadas serán modificadas o no con sólo mirar a la llamada de función.
Los punteros de función: los punteros no son sólo para los objetos en la memoria; también se pueden utilizar para las funciones, permitiendo así una función que se pasa como un parámetro a otra función.
Confusión puntero: los punteros son a menudo acompañados por una sintaxis oscura, y por convención de la denominación de variables consistentes pueden ayudar a que su código sea más legible. Por ejemplo, las convenciones de nombres comunes incluyen el uso de "p" o "ptr" como un prefijo a los nombres de variable puntero.

7) Dentro de lo que es la planificación y la organización que tiene un desarrollador en sí para llevar a cabo un proyecto, ¿que se entiende por la regla del 80-20?

La regla del 80-20 mantiene que cuando construimos software, podemos desarrollar el 80% de lo que fue requerido en el 20% del tiempo. Esta regla es útil tenerla en cuenta ya que en el caso de los programadores, si planean bien la construcción de su programa pueden utilizar el 80% del tiempo sobrante en nuevas prioridades que sirvan para el mejoramiento del software que está creando.

8) ¿Qué se entiende por “grid computing" y cómo lo aplicaría en bioinformática?

Grid en inglés es un término que hace alusión a la distribución de energía, en computación se utiliza para hacer referencia a repartir el procesamiento de datos en diferentes computadoras conectadas, es un concepto similar a una supercomputadora.
Es muy común su uso en bioinformática debido a la cantidad de datos a procesar como sería en el caso de secuenciación de genomas de gran tamaño