**Laboratorio Nro. 4  
Árboles**

## Objetivos

1. Escoger la estructura de datos apropiada para modelar un problema dado
2. Solucionar problemas del mundo real con algoritmos

## 

**Consideraciones Iniciales**

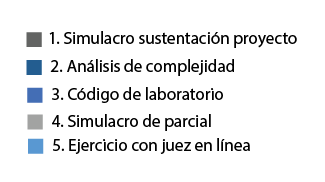
|  |  |
| --- | --- |
| guia-01**Leer la Guía** | Antes de comenzar a resolver el presente laboratorio, leer la ***“Guía Metodológica para la realización y entrega de laboratorios de Estructura de Datos y Algoritmos”*** que les orientará sobre los requisitos de entrega para este y todos los laboratorios, las rúbricas de calificación, el desarrollo de procedimientos, entre otros aspectos importantes. |
| **Registrar Reclamos**  boton-de-preguntas-frecuentes | En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de [**http://bit.ly/2g4TTKf**](http://bit.ly/2g4TTKf), el cual será atendido en la menor brevedad posible. |
| **Traducción de Ejercicios**  Traducción-01 | En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea. |
| **Visualización de Calificaciones**  **examen-01** | A través de ***Eafit Interactiva*** encontrarán **un enlace** que les permitirá **ver un registro de las calificaciones** que **emite el docente** para cada taller de laboratorio y según las rubricas expuestas. ***Véase sección 3, numeral 3.7.*** |
| **GitHub** | Crear un repositorio en su cuenta de GitHub con el nombre st0245. **2.** Crear una carpeta dentro de ese repositorio con el nombre laboratorios. **3.** Dentro de la carpeta laboratorios, crear una carpeta con nombre lab04. **4.** Dentro de la carpeta lab04, crear tres carpetas: informe, codigo y ejercicioEnLinea. **5.**Subir el informe pdf a la carpeta infome, el código del ejercicio 1 a la carpeta codigo y el código del ejercicio en línea a la carpeta ejercicioEnLinea. Así:  st0245-suCodigoAqui  laboratorios  lab01  informe  codigo  ejercicioEnLinea  lab02  ... |

**Intercambio de archivos**

Los archivos que **ustedes deben entregar** al docente son: **un archivo PDF** con el informe de laboratorio usando la plantilla definida, y **dos códigos**, uno con la solución al numeral 1 y otro al numeral 2 del presente. Todo lo anterior se entrega en **GitHub**.



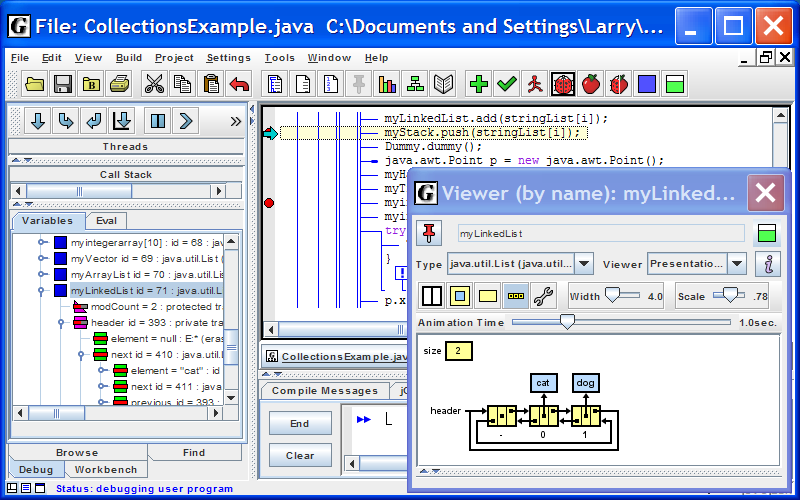
**Porcentajes y criterios de evaluación para el laboratorio**



## 

**Resolver Ejercicios**

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:** Para este taller, de manera especial, se recomienda utilizar el *IDE Jgrasp* **(**[**http://www.jgrasp.org/**](http://www.jgrasp.org/)) porque tiene un depurador gráfico excelente para estructuras de datos |
| alerta-simbolo-dibujado-a-mano | **NOTA 2:**Véase a continuación gráfica de Jgrasp para efectos de ejemplificación |

****

1. **Código para entregar en GitHub:**

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **En la vida real, la documentación del software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126** |

|  |  |  |  |
| --- | --- | --- | --- |
| **guia-01** | Véase Guía ***en Sección 3, numeral 3.4*** |  | *Código de laboratorio en* ***GitHub.*** Véase Guía en **Sección 4, numeral 4.24** |
| **No HTML** | Es opcional entregar documentación. Si lo hace, utilice **Javadoc** o equivalente. No suba el HTML a GitHub. | **no rar** | ***No******se reciben*** *archivos en* ***.RAR*** *ni en* ***.ZIP*** |

**1.1** Los avances en la tecnología han hecho que hoy en día sea necesario almacenar y procesar grandes volúmenes de información. Para poder manejar la información en el disco duro se utiliza un sistema de archivos.

Un sistema de archivos es un componente de un sistema operativo encargado de asignar espacio a los archivos, administrar el espacio libre y permitir el acceso a los datos guardados. Hoy en día existe la necesidad de almacenar y acceder a un número muy grande de archivos en un sistema de archivos. Una de la funciones del sistema de archivos es listar los contenidos de un directorio eficientemente.

**FLECHITA**

El problema de listar el contenido de un directorio consiste **en encontrar eficientemente, los archivos y subdirectorios que se encuentran en un directorio.**

El objetivo es desarrollar una estructura de datos para representar los directorios y archivos en un sistema de archivos, y permitir consultar eficientemente los archivos y subdirectorios que se encuentren en un directorio.

**Ejemplo**

Como un ejemplo, considere la siguiente estructura de directorios en un sistema de archivos con el formato [usuario tamaño] archivo o directorio:

Proyecto/

├── [root 4.0K] DataSets

│   └── [mauriciotoro 252K] treeEtc.txt

├── [mauriciotoro 4.0K] Plantillas

│   ├── [mauriciotoro 463K] ED1-Guia-Proyecto-Entrega.doc

│   ├── [mauriciotoro 1.5M] ED1-Plantilla-Eafit.pptx

│   ├── [mauriciotoro 255K] ejemplo-de-un-reporte-con-plantilla-ACM.pdf

│   ├── [mauriciotoro 1.1M] plantilla-ACM-en-Latex.zip

│   ├── [root 1.2M] plantilla-ACM-en-Word.doc

│   └── [mauriciotoro 4.0K] SIG Proceedings Template-Jan2015 Zip

│   ├── [mauriciotoro 9.4K] acmcopyright.sty

│   ├── [mauriciotoro 144K] acm-update.pdf

│   ├── [mauriciotoro 336K] flies.eps

│   ├── [mauriciotoro 151K] fly.eps

│   ├── [mauriciotoro 3.4K] rosette.eps

│   ├── [mauriciotoro 65K] sig-alternate-05-2015.cls

│   ├── [mauriciotoro 124K] sig-alternate-guide.docx

│   ├── [mauriciotoro 572K] sig-alternate-guide.pdf

│   ├── [mauriciotoro 255K] sig-alternate-sample.pdf

│   ├── [mauriciotoro 26K] sig-alternate-sample.tex

│   └── [mauriciotoro 1.5K] sigproc.bib

└── [mauriciotoro 571K] Proyecto Final ED1 Sistema-directorios Vr 2.0.pdf

Si el usuario consulta los subdirectorios y archivos que se encuentran en “Proyecto/DataSets” la respuesta debe ser “treeEtc.txt”.

Como otro ejemplo, si el usuario consulta los archivos y subdirectorios que se encuentran en “Proyecto/Plantillas/” y ocupan más de 1 Megabyte, la respuesta debe ser “ED1-Plantilla-Eafit.pptx” y “plantilla-ACM-en-Latex.zip”.

Como otro ejemplo, si el usuario consulta los archivos y subdirectorios que se encuentran en “Proyecto/” cuyo dueño sea el usuario root, debe retornar “DataSets/” y “plantilla-ACM-en-Word.doc”

Finalmente, si el usuario consulta por los subdirectorios y archivos que se encuentran en “Proyecto/Datos3” la respuesta debe ser “No such file or directory”.

**Conjuntos de Datos**

La estructura de directorios se entrega en un archivo formato TXT con el formato que se muestra en el ejemplo, que es el formato que entrega el programa de la línea de comandos *tree* –shaul que está disponible en Windows, Linux y Mac OS.

En el github hay 3 conjuntos de datos: ejemplito.txt, juegos.txt y treeEtc.txt

**1.2 [Ejercicio opcional]** **Investiguen cuáles son sus nombres, los nombres de sus padres, los nombres de sus abuelos y los nombres de sus bisabuelos. Con esa información, construyan sus árboles genealógicos en Java usando la clase BinaryTree.** Como un ejemplo, este sería el mío:



|  |  |
| --- | --- |
| ***Vida  Real-01*** | **En la vida real, los árboles de ancestros son utilizados en Filogenética. Filogenética es la parte de la biología que se ocupa de determinar las relaciones evolutivas entre diferentes grupos de organismos** |

Si no conocen el nombre de uno de ellos, no lo escriban. No modelen hermanos ni tíos. Modelen en el método main, como hijos derechos, los hombres y, como hijos izquierdos, las mujeres. La raíz es “Mauricio”.

**1.3 [Ejercicio opcional]** Wilkenson recolectó el árbol familiar de su familia con mucho cariño para dárselo a su abuela Eustaquia. Wilkenson, el nieto de Eustaquia, lo ingresó a *Java* usando las clases Node y BinaryTree. **Escriban un método para la clase BinaryTree que calcule quién es la abuela materna de una persona. No escriban el método main.**

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:** El método debe funcionar para cualquier árbol genealógico, no solo para el Wilkenson. Tenga en cuenta que, en un árbol genealógico, para algunas personas, no se conoce la abuela materna |
| alerta-simbolo-dibujado-a-mano | **NOTA 2:** Considere que los ancestros paternos van a la derecha y los ancestros maternos a la izquierda. |

**2) Ejercicios en línea sin documentación en GitHub**

|  |  |  |  |
| --- | --- | --- | --- |
| **guia-01** | Véase Guía en ***Sección 3, numeral 3.3*** | **No HTML** | **No entregar** documentación **HTML** |
| **C:\Users\Luisa Alzate\AppData\Local\Microsoft\Windows\INetCache\Content.Word\java.png** | *Entregar un archivo en* ***.JAVA*** | **no pdf** | ***No******se reciben*** *archivos en* ***.PDF*** |
|  |  |  | *Código del ejercicio en línea en* ***GitHub.*** Véase Guía en **Sección 4, numeral 4.24** |

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:** Recuerden que, **si toman la respuesta de alguna fuente**, deben **referenciar** según el **tipo de cita** correspondiente***.*** Véase *Guía en* ***Sección 4, numerales 4.16 y 4.17*** |

**2.1 Resuelvan el siguiente problema usando árboles binarios**

|  |
| --- |
| Un árbol de búsqueda binario (o árbol binario de búsqueda) es un árbol binario que satisface las siguientes propiedades:   1. El subárbol izquierdo de un nodo contiene sólo nodos con valores menores al valor del nodo. 2. El subárbol derecho de un nodo contiene sólo nodos con valores mayores al valor del nodo. 3. Tanto el subárbol izquierdo como el subárbol derecho tienen que ser también árboles de búsqueda binarios.     **Figura 1. Ejemplo de un árbol de búsqueda binario**  El recorrido en pre-orden (Raíz – Izquierda - Derecha) imprime los nodos visitando la raíz del árbol, luego recorriendo el subárbol izquierdo y luego recorriendo el subárbol derecho.  El recorrido en pos-orden (Izquierda – Derecha - Raíz) imprime el subárbol izquierdo primero, luego el subárbol derecho y por último la raíz. Por ejemplo, los resultados del recorrido en pre-orden y pos-orden del árbol de búsqueda binario mostrado en la Figura 1 son:   1. **Pre-orden**: 50 30 24 5 28 45 98 52 60 2. **Pos-orden**: 5 28 24 45 30 60 52 98 50   Dado el recorrido en pre-orden de un árbol de búsqueda binario, su tarea es encontrar el recorrido en pos-orden del árbol.  **Entrada**  El valor de cada nodo del árbol de búsqueda binario se dará de acuerdo a un recorrido en pre-orden. Cada nodo tiene un valor que es un entero positivo menor que 106. Todos los valores se dan en líneas separadas (un entero por línea). Ud. puede asumir que un árbol de búsqueda binario no contiene más de 10 000 nodos y que no hay nodos duplicados (es decir, el valor de cada nodo es único en el árbol de búsqueda binario dado).  **Salida**  La salida contiene el resultado del recorrido en pos-orden del árbol de búsqueda binario dado en la entrada. Impriman un valor por línea.  **Ejemplo de entrada**  50  30  24  5  28  45  98  52  60  **Ejemplo de salida**  5  28  24  45  30  60  52  98  50 |

***2.2* [Ejercicio Opcional]Resuelvan el siguiente problema** [**http://bit.ly/2iNuHaa**](http://bit.ly/2iNuHaa)

**3) Simulacro de preguntas de sustentación de Proyectos**

|  |  |  |  |
| --- | --- | --- | --- |
| **guia-01** | Véase Guía en ***Sección 3, Numeral 3.5*** | **C:\Users\Luisa Alzate\AppData\Local\Microsoft\Windows\INetCache\Content.Word\pdf-simbolo-de-formato-de-archivo.png** | *Entregar informe de laboratorio*en **PDF** |
| **C:\Luisa Alzate\2. Laboral\Informal\Mauricio Toro\Iconos y Logos\papel.png** | *Usen la* ***plantilla*** *para responder laboratorios* | ***ICONTECr*** | ***No*** *apliquen* ***Normas Icontec*** *para esto* |

**3.1 ...**

**3.? [ejercicio opcional** **¿Se puede implementar más eficientemente un árbol genealógico para que la búsqueda e inserción se puedan hacer en tiempo logarítmico? ¿O no se puede? ¿Por qué?**

***3.2*** **Expliquen con sus propias palabras cómo funcionan los ejercicios de los numerales 2.1 y 2.2**

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:** Recuerden que debe explicar su implementación en el informe PDF |

**3.3** **Calculen la complejidad de los ejercicios realizados en los numerales 2.1 y 2.2 y agréguenla al informe PDF**

**3.4** **Expliquen con sus palabras las variables (qué es ‘n’, qué es ‘m’, etc.) del cálculo de complejidad del numeral 3.3**

**4) Simulacro de Parcial en el informe PDF**

|  |  |  |  |
| --- | --- | --- | --- |
| **hoja-de-papel-01** | Para este simulacro, agreguen ***sus respuestas* en el informe PDF.**  Resuelva, como mínimo, los ejercicios marcados con color rojo. | **ordenador-peral-01** | *El* ***día del Parcial no tendrán computador, JAVA o acceso a internet.*** |

|  |
| --- |
| **1.** El nodo de un árbol binario se define así:    class Nodo {  Nodo izq;  Nodo der;  int dato;  }  Kefo ha dañado a Dayla su código para determinar **cuál es la altura máxima de un árbol binario**. Dayla no recuerda como lo había hecho y les pide ayuda para que lo completen.    01 int altura(Nodo raiz){  02 if(raiz == null)  03 return 0;  04 int izq =\_\_\_\_\_\_\_;  05 int der =\_\_\_\_\_\_\_;  06 return Math.max(izq, der);}   1. Completen el espacio en línea 04 **(10%)**   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   1. Completen el espacio en línea 05 ***(*10%)**   **2.** A un árbol binario de búsqueda se le ingresan 9 elementos en este orden: 12, 3, 5, 8, 2, 1, 9, 17. **¿Cuántos nodos hay que recorrer antes de encontrar el número 8?** Se cuenta la raíz, pero no se cuenta el nodo donde está el 8.   1. 2 2. 1 3. 3 4. 5   **3.** Definimos el camino desde la raíz hasta una hoja en un árbol binario como una secuencia de nodos empezando en el nodo raíz y bajando hasta una hoja. Una hoja es un nodo que no tiene hijos. Decimos que un árbol vacío no contiene caminos desde la raíz hasta una hoja.  Como un ejemplo, el siguiente árbol tiene 4 caminos desde la raíz hasta una hoja:  **5**  **/ \**  **4 8**  **/ / \**  **11 13 4**  **/ \ \**  **7 2 1**  Caminos desde la raíz hasta una hoja:   * **Camino 1:** 5 4 11 7 * **Camino 2:** 5 4 11 2 * **Camino 3:** 5 8 13 * **Camino 4:** 5 8 4 1   Para este problema nos interesan las sumas de los elementos en los nodos de esos caminos, por ejemplo, la suma del camino [5,4,11,7] es 27.  Dada la raíz de un árbol binario Nodo a y un entero int suma, **decir si existe un camino desde la raíz hasta una hoja tal que al sumar los valores de los nodos de ese camino la suma sea igual al parámetro suma**.  Retorne falso si no se puede encontrar un camino con esa condición. Utilice la definición de Nodo del punto 1.  Desafortunadamente, al código que hizo Dayla le faltan unas líneas y Kefo está de vacaciones.    1 boolean sumaElCamino(Nodo a, int suma) {  2 if (a == null)  3 return \_\_\_\_\_\_\_\_\_\_;  4 if (a.izq == null && a.der == null)  5 return suma == \_\_\_\_\_\_\_\_\_\_;  6 else  7 return sumaElCamino(\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_)  8 || sumaElCamino(\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_); }   1. Completen el espacio de la línea 03 **(10 %)**   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   1. Completen el espacio de la línea 05 **(10 %)**   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   1. Complete los espacios de la línea 07 **(10 %)**   \_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_   1. Complete los espacios de la línea 08 **(10 %)**   \_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_  **4.** Consideren la siguiente definición de árbol binario:    class Node {  public Node left;  public Node right;  public String data;  public Node(String d) {  data = d;  }  }  El siguiente algoritmo imprime todos los valores de un árbol en pre orden.    01 private void printAUX(Node node) {  02 if (node != null) {  03 System.out.println(node.data);  04 printAUX(node.left);  05 printAUX(node.right);  06 }  07 }  08 public boolean print() {  09 printAUX(root);  10 }  **4.1** ¿Cuál ecuación de recurrencia que describe el número de instrucciones que ejecuta el algoritmo print en el peor de los casos?  La variable *n* representa el número de elementos del árbol.   1. *T*(*n*)=*T*(*n*−1)+*C* 2. *T*(*n*)=2.*T*(*n*−1)+*C* 3. *T*(*n*)=2.*T*(*n*/2)+*C* 4. *T*(*n*)=*T*(*n*/2)+*C* 5. *T*(*n*)=*T*(*n*+1)+*C*   **4.2** ¿Cuál es su complejidad asintótica en el peor de los casos del algoritmo print? La variable *n* representa el número de elementos del árbol.   1. *O*(*n*) 2. O(n2) 3. (*log* *n*) 4. *O*(*n*.*m*) 5. *O*(1)   **4.3** ¿Cuál es la salida del algoritmo print para el siguiente árbol?  Tenga en cuenta que la raíz es Wilkenson.  Después de hacer la rotación, usted entenderá que realmente las mujeres van a la izquierda y los hombres a la derecha.  Tenga en cuenta que, en un árbol genealógico, para algunas personas, no se conoce la mamá o el papá.       1. Wilkenson, Sufranio, Piolín, Usnavy, Piolina, Wilberta, Joaquina, Eustaquio, Florinda, Eustaquia, Yovín 2. Sufranio, Piolina, Wilberta, Piolín, Usnavy, Joaquina, Estaquia, Florinda, Wilkenson, Yovín, Eustaquio 3. Wilkenson, Yovín, Eustaquio, Sufranio, Piolina, Wilberta, Piolín, Usnavy, Joaquina, Estaquia, Florinda 4. Wilkenson, Joaquina, Eustaquia, Florinda, Eustaquio, Jovín, Sufranio, Piolina, Wilberta, Piolín, Usnavy 5. Sufranio, Piolina, Wilberta, Piolín, Usnavy, Florinda, Wilkenson, Yovín, Eustaquio, Joaquina, Estaquia   **4.4** ¿Qué modificación hay que hacer algoritmo print para que arroje la siguiente respuesta para el árbol anterior?:  Usnavy, Piolín, Wilberta, Piolina, Sufranio, Florinda, Eustaquio, Yovín, Eustaquia, Joaquina, Wilkenson.  Tenga en cuenta que la raíz es Wilkenson. Como Wilkenson es la raíz, para poder entender el árbol, recomendamos rotarlo 180 grados.  Después de hacer la rotación, usted entenderá que realmente las mujeres van a la izquierda y los hombres a la derecha.  Tenga en cuenta que, en un árbol genealógico, para algunas personas, no se conoce la mamá o el papá.   1. Cambiar el orden de las líneas 03, 04 y 05 por 05, 04, 03 2. Cambiar el orden de las líneas 03, 04 y 05 por 04, 05, 03 3. Cambiar el orden de las líneas 03, 04 y 05 por 03, 05, 04 4. Cambiar el orden de las líneas 03, 04 por 06, 07 5. Intercambiar la línea 03 y la línea 04   **5.** Luis escribió un programa para insertar un número en un árbol binario de búsqueda. En dicho árbol, él quiere tener los números menores o iguales a la raíz a la derecha y los mayores a la raíz a la izquierda.  Ayúdele a completar su código. El algoritmo recibe la raíz de un árbol p y un número a insertar toInsert, y retorna la raíz del árbol con el elemento insertado donde corresponde.  01 private Node insert(Node p,int toInsert){  02 if (p == null)  03 return new Node(toInsert);  04 if (...........)  05 return p;  06 if (...........)  07 p.left = insert(p.left, toInsert);  08 else  09 p.right = insert(p.right, toInsert);  10 return p;  11 }   1. Complete, por favor, la línea 4 con la condición que corresponde **(10 %)**   ..............   1. Complete, por favor, la línea 6 con la condición que corresponde **(10 %)**   ..............  **6.** Dado un árbol *n*−*ario*, un camino desde la raíz a cualquiera de sus hojas se considera **simple** si la suma de sus elementos pares es igual a la suma de sus elementos impares.  Por ejemplo, un camino desde la raíz hasta una hoja con los elementos [1, 2, 1] es **simple**, pero el camino desde la raíz hasta una hoja [1, 3, 1] **no es simple**.  Su tarea es determinar cuántos caminos **simples** hay en un árbol. La implementación de un nodo *n*−*ario* es la siguiente:  class NNodo{  int val; //Valor en el nodo actual.  //Hijos del nodo actual.  LinkedList<NNodo> hijos;  }    **6.1** ¿Cuantos caminos **simples** hay desde una raíz hasta una hoja en el árbol anterior?  **a**. 1  **b.** 2  **c.** 3  **d.** 4  El siguiente código nos ayuda a determinar el número de caminos simples en un árbol, pero faltan algunas líneas. Por favor, complétenlas.    01 public int cuantosSimples(NNodo raiz, int suma){  02 //Arbol vacio  03 if(raiz == null)  04 \_\_\_\_\_\_\_\_\_\_\_\_\_\_;  05 //Hoja  06 if(raiz.hijos.size() \_\_\_\_\_\_\_) // Si suma es 0  07 return (suma == 0) ? 1 : 0; // Retorne 1, sino, 0  08 int total = 0;  09 for(NNodo n: raiz.hijos)  10 if(n.val % 2 == 0) // Par  11 total += cuantosSimples(n, suma + n.val);  12 else // Impar  13 total += cuantosSimples(n, suma - n.val);  15 return total;  16 }  17  18 public int cuantosSimples(NNodo raiz){  19 int val = (raiz.val % 2 == 0) ?  raiz.val : -raiz.val;  20 return cuantosSimples(raiz, val);  21 }  **6.2** Completen la línea 4  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **6.3** Completen la línea 6  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **7.** Los siguientes algoritmos son el recorrido en in orden y pos orden. En la vida real, estos recorridos son de utilidad para hacer procesamiento del lenguaje natural.  void InOrden(Node node){  if (node != null){  InOrden(node.left);  System.out.println(node.data);  InOrden(node.right);  } }  void PosOrden(Node node){  if (node != null){  PosOrden(node.left);  PosOrden(node.right);  System.out.println(node.data);  } }  Consideren el recorrido in orden y pos orden de un **Árbol binario de búsqueda** con los siguientes elementos 4, 9, 1, 5, 7, 11, 13, 2, 0, 10. Ahora, la impresión del recorrido in-orden nos devolverá los elementos, y, la impresión del recorrido pos-orden devolverá .    **7.1** ¿Cuál es la impresión pos-orden?  **1.** 0, 2, 1, 7, 5, 10, 13, 11, 9, 4  **2.** 0, 1, 2, 4, 5, 7, 9, 10, 11, 13  **3.** 0, 2, 1, 7, 10, 5, 13, 11, 9, 4  **4.** 4, 1, 0, 2, 9, 5, 7, 11, 13, 10  **7.2** Supongan que hacemos la comparación elemento a elemento del recorrido en in-orden y el recorrido en pos-orden. ¿Cuántos elementos aparecen en la misma posición en ambos recorridos? Como uno ejemplo, si un recorrido es 1,3,4,2 y el otro es 2,3,4,1, la respuesta es 2 porque el 3 y el 4 aparecen en la misma posición en ambos.  **1.** 7  **2.** 2  **3.** 5  **4.** 8 Considere el siguiente árbol para los ejercicios 8,9 y 10   **8.** Sean *A* y *B* las salidas de los recorridos pre-orden y pos-orden del árbol binario anterior, respectivamente. Determine el número de elementos para los cuales se cumple que para 1≤*i*≤8.  (a)3  (b) 2  (c) 4  (d) 0  **9.** ¿Cuál es la salida del recorrido in-orden del árbol binario anterior?  (a)5, 3, 6, 1, 7, 4, 8, 0, 2  (b) 0, 1, 3, 5, 6, 4, 7, 8, 2  (c) 5, 6, 3, 7, 8, 4, 1, 2, 0  (d) 5, 6, 3, 1, 7, 4, 8, 0, 2  **10.** ¿Es un ***árbol binario de búsqueda*** el árbol anterior?  (a) Sí  (b) No  **11.** El sub-índice *D* indica que el nodo es un hijo derecho con respecto a su padre. El sub-índice *I* indica que es un hijo izquierdo con respecto a su padre.    **11.1.** ¿Cuál es la salida del recorrido en *in-orden* del árbol anterior, tomando el nodo 4 como el nodo raíz?  (a) 3, 2, 0, 7, 5, 6, 4  (b) 2, 3, 4, 0, 5, 7, 6  (c) 4, 2, 3, 6, 0, 5, 7  (d) 4, 7, 3, 6, 2, 0, 5  **11.2.**  Asuma que el nodo 4 es la raíz del árbol binario anterior. Asuma que la salida del recorrido *pre-orden* es {a1,a2,a3,...,an} y la salida del recorrido *in-orden* es {b1,b2,b3,...,bn} . ¿Cuál es el primer valor de *i* para el cual ai=bi ? Note que la *i* empieza en 1.  (a) 5  (b) 4  (c) 3  (d) 2  Por ejemplo, asuma que las salidas son *a*={4,6,5,0,7,2,3} y *b*={6,4,7,0,5,2,3}. La respuesta sería 4 porque a4 = b4 = 0 e i=4 es el primer valor para el que ai=bi.  **11.3.** ¿Es un ***árbol binario de búsqueda*** el árbol anterior?  (a) Sí  (b) No |

1. **[Ejercicio Opcional] Lectura recomendada**

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **"Quienes se preparan para el ejercicio de una profesión requieren la adquisición de competencias que necesariamente se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades…”**  Tomado de[**http://bit.ly/2gJKzJD**](http://bit.ly/2gJKzJD) |

|  |  |
| --- | --- |
| guia-01 | *Véase Guía en* ***Sección 3, numeral 3.5 y 4.20*** *de la Guía Metodológica, “Lectura recomendada” y “Ejemplo para realización de actividades de las Lecturas Recomendadas*”, respectivamente |

Posterior a la lectura del ***“Robert Lafore, Data Structures and Algorithms in Java (2nd edition), 2002. Chapter 8: Binary Trees.”,*** realicen las siguientes actividades que les permitirán sumar puntos adicionales:

1. Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras
2. Hagan un mapa conceptual que destaque los principales elementos teóricos.

|  |  |  |
| --- | --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:** Si desean una lectura adicional en inglés, consideren la siguiente: ***“Narasimha Karumanchi, Data Structures and Algorithms made easy in Java, (2nd edition), 2011. Section 6.9 BST”,*** que encuentran en biblioteca. | |
| alerta-simbolo-dibujado-a-mano | ***NOTA 2:*** Estas respuestas también deben incluirlas en el informe PDF |  |

1. **[Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual**

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **El trabajo en equipo es una exigencia actual del mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques y trabajes bien con otras personas”**  Tomado de[**http://bit.ly/1B6hUDp**](http://bit.ly/1B6hUDp) |

|  |  |
| --- | --- |
| guia-01 | *Véase Guía en* ***Sección 3, numeral 3.6*** *y* ***Sección 4, numerales 4.21, 4.22*** *y* ***4.23*** *de la Guía Metodológica* |

1. Entreguen copia de todas las actas de reunión usando el tablero Kanban, con fecha, hora e integrantes que participaron
2. Entreguen el reporte de *git*, *svn* o *mercuria*l con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron
3. Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | ***NOTA 1:*** Estas respuestas también deben incluirlas en el informe PDF |

1. **[Ejercicio Opcional] Laboratorio en inglés:**

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **El inglés es un idioma muy importante en la Ingeniería de Sistemas porque la mayoría de los avances en tecnología se publican en este idioma y la traducción, usualmente se demora un tiempo y es sólo un resumen de la información original.**  **Adicionalmente, dominar el inglés permite conseguir trabajos en el exterior que son muy bien remunerados**  **Tomado de goo.gl/4s3LmZ** |

**Entreguen el código y el informe traducido al inglés. Utilicen la plantilla dispuesta en este idioma para el laboratorio**

**Resumen de Ejercicios a Resolver**

**1.1 Investiguen cuáles son sus nombres, los nombres de sus padres, los nombres de sus abuelos y los nombres de sus bisabuelos. Con esa información, construyan sus árboles genealógicos en Java usando la clase BinaryTree**

**1.2** **Escriban un método para la clase BinaryTree que calcule quién es la abuela materna de una persona. No escriban el método main.**

**2.1 Resuelvan el problema usando árboles binarios**

***2.2* [Ejercicio Opcional]Resuelvan el siguiente problema** [**http://bit.ly/2iNuHaa**](http://bit.ly/2iNuHaa)

**3.1** **¿Se puede implementar más eficientemente un árbol genealógico para que la búsqueda e inserción se puedan hacer en tiempo logarítmico? ¿O no se puede? ¿Por qué?**

***3.2*** **Expliquen con sus propias palabras cómo funcionan los ejercicios de los numerales 2.1 y 2.2**

**3.3** **Calculen la complejidad de los ejercicios realizados en los numerales 2.1 y 2.2 y agréguenla al informe PDF**

**3.4** **Expliquen con sus palabras las variables (qué es ‘n’, qué es ‘m’, etc.) del cálculo de complejidad del numeral 3.3**

**4. Simulacro parcial**

**5. [Ejercicio Opcional] Lectura recomendada**

**6. [Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual**

**7. [Ejercicio Opcional] Laboratorio en inglés**

**Ayudas para resolver los ejercicios**

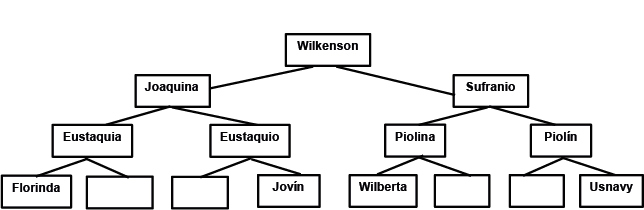
|  |  |
| --- | --- |
| **Ayudas para el Ejercicio 1…………………….** | [**Pág. 26**](#Ejer1) |
| **Ayudas para el Ejercicio 1.3…………………….** | [**Pág. 26**](#Ejer13) |
| **Ayudas para el Ejercicio 2.1……………………..** | [**Pág. 26**](#Ejer21) |
| **Ayudas para el Ejercicio 2.2……………………..** | [**Pág. 27**](#Ejer22) |
| **Ayudas para el Ejercicio 3.2.……………………** | [**Pág. 27**](#Ejer32) |
| **Ayudas para el Ejercicio 3.4…………………….** | [**Pág. 27**](#Ejer34) |
| **Ayudas para el Ejercicio 4……………………….** | [**Pág. 27**](#Ejer4) |
| **Ayudas para el Ejercicio 5a……………………...** | [**Pág. 28**](#Ejer5a) |
| **Ayudas para el Ejercicio 5b……………………...** | [**Pág. 28**](#Ejer5b) |
| **Ayudas para el Ejercicio 6a……………………...** | [**Pág. 28**](#Ejer6a) |
| **Ayudas para el Ejercicio 6b……………………...** | [**Pág. 28**](#Ejer6b) |
| **Ayudas para el Ejercicio 6c……………………...** | [**Pág. 28**](#Ejer6c) |

**Ayudas para el** **Ejercicio 1**

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA3:** Si deciden hacer la documentación de los puntos del numeral 1, vean la *Guía en* ***Sección 4, numeral 4.1*** *“Cómo escribir la documentación HTML de un código usando JavaDoc”* |

**Ayudas para el** **Ejercicio 1.3**

|  |  |
| --- | --- |
| clave | **PISTA 1:** Como un ejemplo, en el árbol genealógico de Wilkenson, Eustaquia es la abuela materna de Wilkenson. Otro ejemplo, Wilberta es la abuela materna de Sufranio y no se conoce la abuela materna de Eustaquia. |
| clave | **PISTA 2:** Utilicen el siguiente estereotipo para el método:  *public String getGrandMothersName(String name)* |
| clave | **PISTA 3:** Retornen una cadena vacía si no se encuentra el nombre de la persona *name* o la abuela materna de *name*. A continuación, un ejemplo de un árbol genealógico: |

******

**Ayudas para el** **Ejercicio 2.1**

|  |  |
| --- | --- |
| clave | **PISTA 1:** Léase sobre [**http://bit.ly/1V5RqKL**](http://bit.ly/1V5RqKL) |
| clave | **PISTA 2:** No es posible reconstruir un árbol binario, en general, con un el recorrido pre-orden porque habría muchas posibles respuestas. Para poder hacer este ejercicio hay que recordar que es un árbol binario de búsqueda, es decir, donde los menores van a la izquierda y los mayores van a la derecha. |

**Ayudas para el** **Ejercicio 2.2**

|  |  |
| --- | --- |
| clave | **PISTA 1:** Use *búsqueda en profundidad (pre orden, in orden o pos orden)*. Es similar a lo que hacen en Lenguajes de Programación para construir el árbol sintáctico en una expresión aritmética con paréntesis. |

**Ayudas para el Ejercicio 3.2**

|  |  |
| --- | --- |
| clave | **PISTA 1:** Léase [**http://bit.ly/1V5RqKL**](http://bit.ly/1V5RqKL) |
| clave | **PISTA 2:** Léase [**http://bit.ly/2i8DgM4**](http://bit.ly/2i8DgM4) |

**Ayudas para el** **Ejercicio 3.4**

|  |  |
| --- | --- |
| clave | **PISTA 1:** Véase***Guía en Sección 4, numeral 4.11*** *“Cómo escribir la complejidad de un ejercicio en línea”* |
| clave | **PISTA 2:** Véase***Guía en Sección 4, numeral 4.19 “****Ejemplos para calcular la complejidad de un ejercicio de CodingBat”* |

**Ayudas para el Ejercicio 4**

|  |  |
| --- | --- |
| clave | **PISTA 1:** *Véase* ***Guía en Sección 4, Numeral 4.18*** “Respuestas del Quiz” |
| clave | **PISTA 2:** Lean las diapositivas tituladas *“****Data Structures I: Binary trees”,*** encontrarán la mayoría de las respuestas |

**Ayudas para el** **Ejercicio 5a**

|  |  |
| --- | --- |
| clave | **PISTA 1:** En el siguiente enlace, unos consejos de cómo hacer un buen resumen[**http://bit.ly/2knU3Pv**](http://bit.ly/2knU3Pv) |
| clave | **PISTA 2:** [**Aquí**](http://bit.ly/28JXqIG) le explican cómo contar el número de palabras en Microsoft Word |

**Ayudas para el** **Ejercicio 5b**

|  |  |
| --- | --- |
| clave | **PISTA 1:** Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en [**https://cacoo.com/**](https://cacoo.com/)o **[https://www.mindmup.com/#m:new-a-1437527273469](https://www.mindmup.com/" \l "m:new-a-1437527273469)** |

**Ayudas para el** **Ejercicio 6a**

|  |  |
| --- | --- |
| clave | **PISTA 1:** *Véase* ***Guía en Sección 4, Numeral 4.21*** “*Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban”* |

**Ayudas para el** **Ejercicio 6b**

|  |  |
| --- | --- |
| clave | **PISTA 1: *Véase*** *Guía en Sección 4, Numeral 4.23*“**Cómo generar el historial de cambios en el código de un repositorio que está en svn”** |

**Ayudas para el** **Ejercicio 6c**

|  |  |
| --- | --- |
| clave | **PISTA 1: *Véase*** *Guía en Sección 4, Numeral 4.22* “***Cómo ver el historial de revisión de un archivo en Google Docs*”** |