|  |  |
| --- | --- |
| **Prueba 3**  1S - 2015 | NOMBRE:  NRO.MATRICULA :  ☐ **Estructura de Datos** ☐ **Complejidad Computacional** |

# Recursión

|  |  |
| --- | --- |
| 1. En base el método  *dibujaCirculo(x, y, radio)* que dibuja un circulo de radio *r* en el punto *x,y*.  Escriba un método recursivo que genere el dibujo de la derecha. [6 ptos.] | Macintosh HD:Users:patricio:Downloads:circulos.png |

1. Nombre la función matemática implementa el siguiente método. [*2 ptos*.]

**int A(int i) {**

**if( i == 1)**

**return 1;**

**else**

**return(i\*(A(i-1)));**

**}**

# Árboles Binarios

1. Un árbol binario es un árbol de búsqueda cuando [*2 ptos.*]
   1. Cada nodo (no hoja) tiene hijos cuyos valores son menores o iguales que sus padres.
   2. En la ruta desde la raíz a cada nodo hoja, el valor de cada nodo es mayor o igual que el valor del padre.
   3. Un nodo puede tener un máximo de dos hijos.
   4. Todo hijo izquierdo contiene un valor menor que su padre y cada hijo derecho contiene un valor mayor o igual que su padre.
   5. Ninguna de las anteriores
2. En un árbol binario completo con 20 nodos, ¿cuántos nodos hay en el nivel 4 considerando  
   a la raíz como el nivel 0? [*2 ptos.*]
3. En la siguiente clase hay un error, explique cual es este error. [*2 ptos.*]

|  |  |
| --- | --- |
|  |  |

1. Un árbol desbalanceado es aquel donde [1 *ptos.*]  
   1. La mayoría de sus nodos tiene valores mayores que la media.
   2. El comportamiento de los nodos es impredecible.
   3. Los nodos generan una forma de paraguas.
   4. En donde la raíz o cualquier otro nodo tiene muchos más hijos izquierdos que derechos o viceversa.
2. ¿Cómo se puede generar un árbol binario desbalanceado? [2 *ptos.*]

1. Eliminar un nodo con un solo hijo de un árbol de búsqueda binario requiere encontrar al sucesor del nodo. [*1 pto.*]  
   1. Verdadero.
   2. Falso.
2. Identifique el método asociado al código de la columna de la izquierda. [*3 ptos.*]

|  |  |
| --- | --- |
| **public** Node A(**int** key) {  Node current = root;  **while**(current.iData != key {  **if**(key < current.iData)  current = current.leftChild;  **else**  current = current.rightChild;  **if**(current == **null**) **return** **null**;  }  **return** current;  } |  |
| **public** **void** B(**int** id) {  Node newNode = **new** Node();  newNode.iData = id;  **if**(root==**null)** root = newNode;  **else** {  Node current = root;  Node parent;  **while**(**true**) {  parent = current;  **if**(id < current.iData) {  current = current.leftChild;  **if**(current == **null**) {  parent.leftChild = newNode;  **return**;}  }  **else** {  current = current.rightChild;  **if**(current == **null**) {  parent.rightChild = newNode;  **return**;}  }  }  }  } |  |

1. Escriba el método para calcular la diferencia entre el *valor máximo* y el *valor mínimo* de un árbol de búsqueda binario. [6 *ptos.*]
2. Dibuje el árbol binario resultante cuando se elimina el nodo “71” [*2 ptos.*]



1. Suponga que Ud. quiere generar un árbol de búsqueda binario balanceado, ¿qué estrategia de ingreso de datos utilizaría?   
   [2 *ptos*.]