ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
ESTRUCTURAS DE DATOS
SEGUNDA EVALUACIÓN - II TÉRMINO 2016

Nombre: _____ ***Matrícula:*** _____

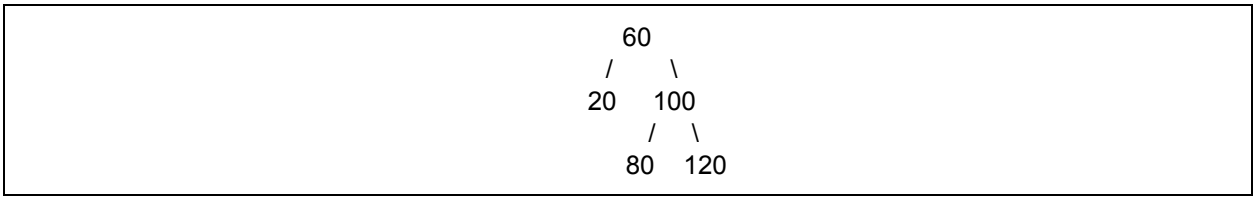
TEMA 1 (15 puntos)

Seleccione la respuesta correcta y **justifique**

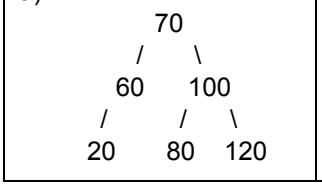
1. Para realizar el recorrido en profundidad en un grafo es necesario utilizar:
    a. Una cola
    b. Una pila
    c. Un diccionario
    d. Una cola de prioridad

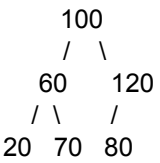
2. Se dispone de una tabla hash de tamaño 12 que utiliza como función hash la aritmética modular y una función **cuadrática** para resolver las colisiones. Se almacenan las claves {29, 41, 22, 31, 50, 19, 42, 38} en las posiciones {5, 6, 10, 7, 2, 8, 3, 4} de la tabla. ¿Cuántas posiciones habrá que examinar para encontrar la clave 42 (contar también la posición donde finalmente se encuentra la clave 42)?
    a. 1
    b. 3
    c. 4
    d. 5

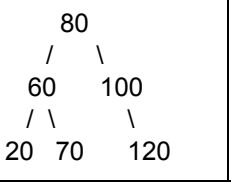
3. Considere el siguiente árbol AVL:

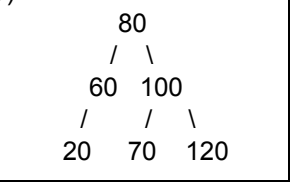


Cual es el árbol resultante después de insertar 70? Justifique con la rotación a utilizar.

a)


b)


c)


d)


4. Se tiene un árbol binario perfecto (todos los nodos internos tienen dos hijos) que almacena caracteres en los nodos. El recorrido del árbol en orden produce la siguiente secuencia de elementos:

   B D E A F G C

   Indicar cuál de entre las siguientes secuencias de elementos es la que produce el recorrido en postorden de dicho árbol:
   a. B E D F C G A
   b. A B C D E F G
   c. C E D F G E A
   d. C G F A E D B

5. Cual es la máxima altura que puede tener un árbol (AVL) con 7 nodos? Recuerde: un árbol con un solo nodo tiene altura 1
   a. 3
   b. 4
   c. 5
   d. 6

TEMA 2 (16 puntos)

Considere las siguientes estructuras:

HashMap	Grafo Dirigido sin ciclos
Grafo No Dirigido	Árbol Binario
Grafo Dirigido con ciclos	Heap

Seleccione y **justifique** la estructura de datos más apropiada para cada uno de los siguientes problemas:

a) El departamento de matemáticas de una institución desea desarrollar una herramienta computacional que permita crear y evaluar expresiones lógicas (AND, OR, NOT) de manera eficiente.

b) El CNE ha decidido implementar una aplicación móvil que permita consultar el lugar de votación de un ciudadano y han decidido que la consulta sea mediante el ingreso de un número de cédula. Se desea que las consultas sea rápidas.

c) El CEEMP desea implementar una aplicación que permita conectar a dos o más estudiantes de distintas facultades que tengan un interés en común (por ejemplo: marketing digital, aplicaciones móviles, etc) para que así puedan lanzar un proyecto en conjunto. Cabe decir, que el CEEMP cuenta con un archivo de todos los estudiantes y sus áreas de interés.

d) El departamento informático de una universidad cuenta con un archivo donde están los nombres y calificaciones de estudiantes, y les han solicitado una lista con todos los estudiantes ordenados por el promedio de calificaciones para la asignación de becas estudiantiles.

TEMA 3 (14 puntos)
Una empresa almacena los registros (logs) de los sitios web que visita el personal en la organización con su respectivo tiempo de conexión. Los registros se encuentran en una lista con el siguiente formato:

```
logs = [
        "user1|www.facebook.com|30",
        "user2|www.google.com|20",
        "user1|www.facebook.com|40",
        "user1|www.twitter.com|20",
         ...
]
```

A usted se le solicita implementar la siguiente función:

```
public HashMap<String,HashMap<String,Integer>> crearMapaVisitas(LinkedList<String> logs)
```

Esta función retorna un mapa con la siguiente estructura:

```
visitas = {
        "User1":{ "www.facebook.com":70, "www.twitter.com":20 },
        "User2":{ "www.google.com":20 },
        ..
}
```

TEMA 4 (25 puntos)
Con las siguientes definiciones del TDA Arbol y Nodo:

```
public class Nodo<T> {
        int clave;
        T contenido;
        Nodo izq, der;
}
```

```
public class Arbol<T> {
        Nodo<T> raiz;
}
```

1. Implemente el método estático **arbolesIguales** que recibe dos árboles y retorna si son iguales. **(10 puntos)**

   ```
   public static boolean arbolesIguales(Arbol arbol1, Arbol arbol2)
   ```

2. Implemente el método público **sumaNivel** que recibe un nivel y devuelve la suma de todas las claves de ese nivel. **(15 puntos)**

   ```
   public int sumaNivel(int nivel)
   private int sumaNivel(Nodo nodo, int nivel, int nivelActual)
   ```

TEMA 5 (30 puntos)

1. El robot Pioneer P3-DX se encuentra en el punto “Inicio” del escenario que se muestra en la Fig. 1, ejecute el algoritmo correspondiente para ayudar al robot a encontrar la ruta más corta para llegar al punto “Fin”. En su respuesta debe detallar el proceso del algoritmo que utilizó para solucionar el problema. **(10 puntos)**

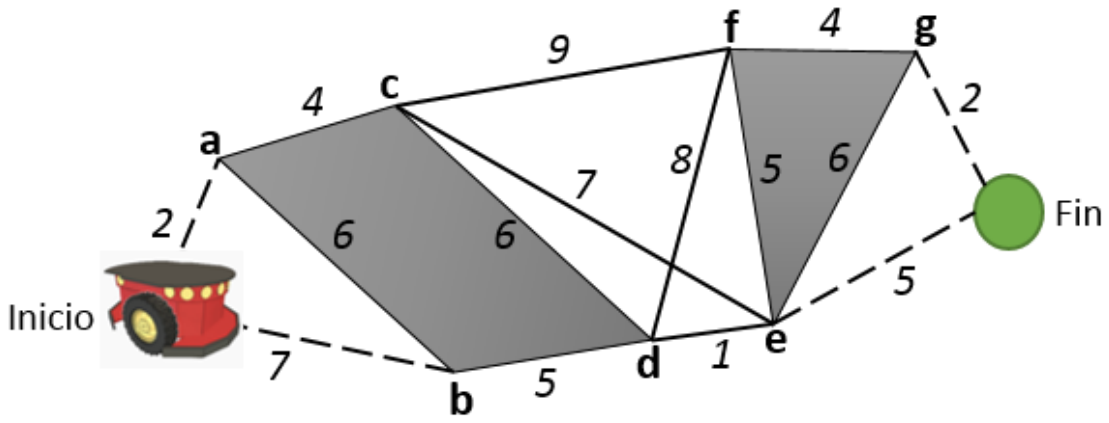
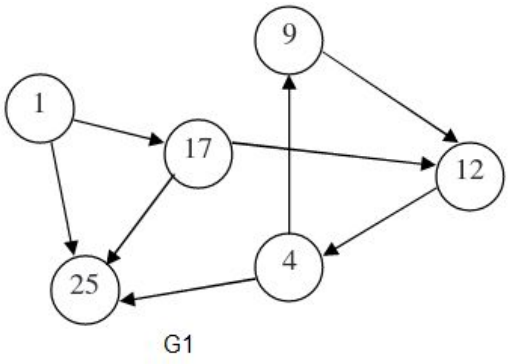


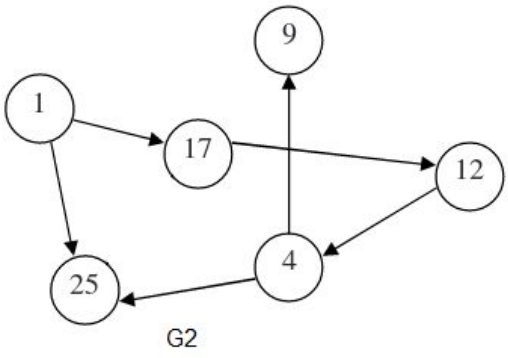
Fig. 1

2. Implemente el método booleano ***acíclico*()**, que indica si un grafo no tiene ningún ciclo. Un ciclo es una sucesión de aristas adyacentes, donde no se recorre dos veces la misma arista, y donde se regresa al punto inicial. **(20 puntos)**



G1

G1.acíclico() retorna false



G2

G2.acíclico() retorna true

Referencias

public class HashMap<K,V>	
HashMap()	Constructor, inicializa el hashmap.
boolean containsKey(Object key)	Retorna true si el map contiene la clave
V get(Object key)	Obtiene el valor asociado con la clave.
V put(K key, V value)	Agrega una entrada al hashmap con la clave y valor pasados como parámetros.
V remove(Object key)	Remueve la clave y el valor del hashmap/
Set<K> keySet()	Retorna un Set con todas las claves.
Set<Map.Entry<K,V>> entrySet()	Retorna un Set con un par clave-valor
Collection<V> values()	Retorna una colección con todos los valores del hashmap.
public class Grafo<T>	
Grafo(boolean dirigido)	Constructor, inicializa el grafo. Si dirigido es true creará un Grafo Dirigido caso contrario será un Grafo No Dirigido.
void agregarVertice(T contenido)	Agrega un nuevo vértice al grafo
void agregarArco(T origen, T destino) void agregarArco(T origen, T destino, int peso)	Agrega un arco entre dos vértices. Si se le especifica el peso, entonces ese será el peso del arco. Caso contrario, el peso será de 1.
List<Vertice<T>> getVertices()	Retorna una lista con todos los vértices del grafo.
Vertice<T> buscarVertice(T contenido)	Retorna el vértice con el contenido especificado, si no existe retorna null.
Arco<T> buscarArco(T origen, T destino)	Retorna true si existe un arco entre origen y destino.
void removerVertice(T contenido)	Remueve el vértice con el contenido especificado, asimismo que todos los arcos que tiene.
void removerArco(T origen, T destino)	Remueve el arco con el origen y destino especificados.
public class Vertice<T>	
Vertice(T contenido)	Constructor

T getContenido()	Obtiene el contenido del vértice
void setContenido(T contenido)	Establece el contenido del vértice
boolean estaVisitado()	Obtiene si el vértice ha sido visitado o no
void setVisitado(boolean visited)	Establece un vértice como visitado
List<Arco<T>> getArcos()	Retorna la lista de arcos del vértice.
public class Arco<T>	
Arco(Vertice origen, Vertice destino, int peso)	Constructor
Vertice<T> getOrigen()	Obtiene el vértice de origen
void setOrigen(Vertice<T> origen)	Establece el vértice de origen
Vertice<T> getDestino()	Obtiene el vértice de destino
void setDestino(Vertice<T> destino)	Establece el vértice de destino
int getPeso()	Obtiene el peso del arco
void setPeso(int peso)	Establece el peso del arco