EIF207 Estructuras de Datos
Examen final

Prof. M.Sc. Georges Alfaro S.

Complete cada uno de los ejercicios solicitados.

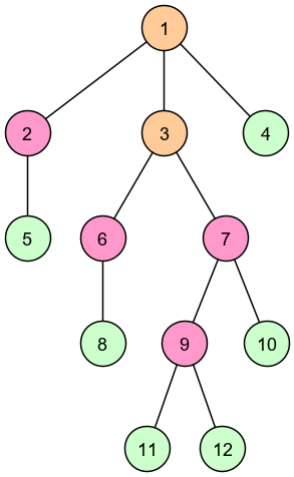
Complete los ejercicios en Java, empleando el IDE NetBeans. Guarde las carpetas de proyecto en una sola carpeta con su nombre en el momento de hacer la entrega. Recuerde que los programas no deberán tener errores de sintaxis. Cualquier ejercicio que no compile correctamente, no será calificado. Debe utilizar la biblioteca de colecciones que se estudió en clase o una implementación propia, pero no deberá usar las colecciones de la biblioteca `java.util` (puede sin embargo usar otras clases que no implementen colecciones, como el `Iterator`, por ejemplo).

PARTE 1

Ejercicio #1 (25 pts.)

Escriba una función que reciba como parámetro dos vértices arbitrarios de un árbol de cualquier orden y encuentre la **mayor distancia** entre ambos vértices.

Considere, por ejemplo, el árbol siguiente:



El nodo 8 tiene los ancestros: 6, 3 y 1. El nodo 12 tiene los ancestros 9, 7, 3 y 1. Los vértices 3 y 1 son ancestros comunes a ambos vértices. El ancestro más cercano en este caso es el nodo **3**. Observe que todos los nodos tienen un conjunto no vacío de vértices comunes, ya que la raíz es un ancestro común para cualquier vértice.

La distancia entre los nodos 8 y 12 es 2 + 3 = 5 (la distancia del nodo 8 hasta el antecesor común es 2, y la distancia desde 12 al mismo vértice es 3). Observe también que un nodo puede ser antecesor del otro. En este caso, la distancia simplemente es la diferencia entre la profundidad de ambos nodos. Por ejemplo, la distancia entre los nodos 3 y 10 es 2 (3 es ancestro de 10).

Ejercicio #2 (25 pts.)

El triángulo de Pascal es una representación de los coeficientes binomiales ordenados en forma de triángulo. Es llamado así en honor al filósofo y matemático francés Blaise Pascal, quien introdujo esta notación en 1654.

Escriba dos funciones: una para **calcular un coeficiente binomial** cualquiera, para dos parámetros enteros *n* y *k*, y un método para **imprimir** en consola el **triángulo de Pascal**.

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

No escriba directamente una función factorial, defina una función recursiva considerando las siguientes identidades:

$$\binom{n}{n} = \binom{n}{0} = 1$$

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$$

Escriba luego un método que imprima el triángulo de Pascal en la consola.

Por ejemplo, al invocar a la función:

```
triangulo(8);
```

el programa mostraría en consola la siguiente salida:

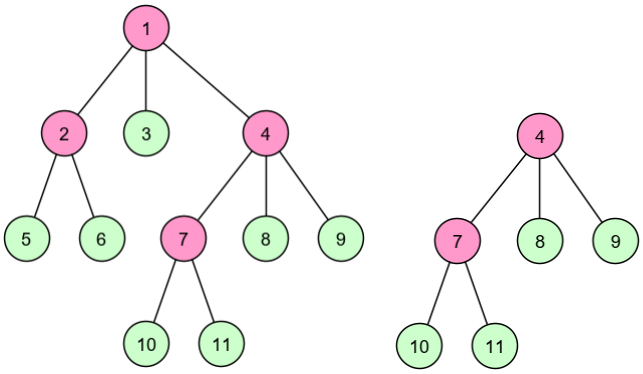
```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28   8   1
```

Observe el sangrado (espaciado) de los resultados, de manera que cada fila aparece centrada en la salida.

Ejercicio #3 (25 pts.)

Escriba una función que indique si un árbol es un subárbol de otro.

Por ejemplo, dados los dos árboles:



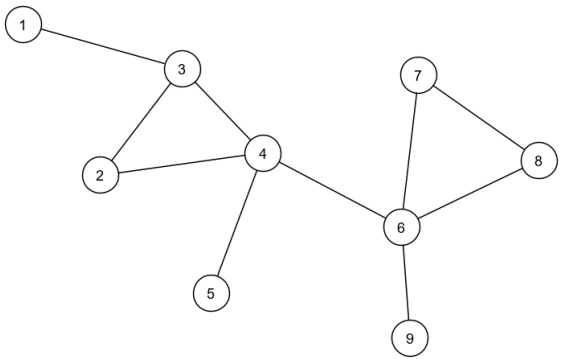
El árbol de la derecha es un subárbol de la izquierda.

Para esto, sobrecargue el método `equals()` para el tipo de elemento del árbol, que compare el valor contenido en ambos vértices y que también compare los subárboles asociados de manera recursiva. Tome en cuenta el orden de los sucesores de cada nodo. Haga lo mismo para la clase que define el tipo del nodo o vértice del árbol.

Ejercicio #4 (25 pts.)

Escriba una función que encuentre todos los grupos de nodos que forman un ciclo en un grafo cualquiera. Recuerde que existe un ciclo cuando es posible encontrar un camino con el mismo nodo de inicio y destino. Si el grafo es no dirigido, un arco no deber ser recorrido dos veces (una vez en cada sentido).

Por ejemplo, en el siguiente grafo:



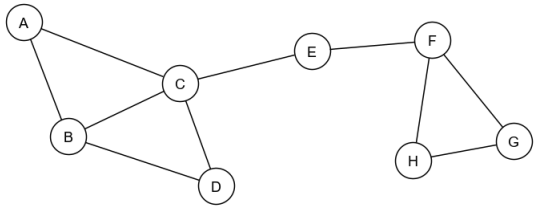
Existen 2 grupos de vértices que forman un ciclo: {{2, 3, 4}, {6, 7, 8}}

La función debe regresar una lista de grupos de vértices (una lista de listas de vértices). Si no existe ningún ciclo, el resultado será una lista vacía.

Ejercicio #5 (15 pts. Opcional)

El **nivel de un vértice** con respecto a otro es la longitud (número de arcos recorridos) del camino más corto entre ambos.

Si considera el siguiente grafo, el nivel entre cada par de vértices está indicado en la tabla.



	A	B	C	D	E	F	G	H
A	0	1	1	2	2	3	4	4
B	1	0	1	1	2	3	4	4
C	1	1	0	1	1	2	3	3
D	2	1	1	0	2	3	4	4
E	2	2	1	2	0	1	2	2
F	3	3	2	3	1	0	1	1
G	4	4	3	4	2	1	0	1
H	4	4	3	4	2	1	1	0

Construya una función que calcule el nivel entre dos vértices y muestre una tabla como la anterior.

Notas

Para todos los ejercicios de grafos, puede utilizar la implementación con listas de adyacencia o usando la matriz de adyacencia..