INFORMÁTICA III
FISICC

PROYECTO I - RECURSIVIDAD

Objetivo: El objetivo del proyecto es reforzar la importancia que tienen las herramientas recursivas en el campo de las Ciencias de la Computación. Con este proyecto se logran asociar los conceptos matemáticos con el área específica de la programación, demostrando lo sencillo que es resolver problemas complejos (ej. hanoi) por métodos recursivos.

Descripción: Este proyecto está enfocado a aplicar las definiciones recursivas expuestas en el tema 1 de clase que incluyen: el factorial de un número, la función de Ackerman y la búsqueda recursiva en estructuras de datos, en este caso un árbol. El programa se desarrollará totalmente en lenguaje Java y usted deberá restringirse a las indicaciones que se presentan más adelante. **El proyecto deberá presentarse en grupos de máximo 3 estudiantes.**

Objetivos Secundarios:

- Familiarizarse en el uso y programación de la ILC (Interfaz de Línea de Comando) la cual se estará utilizando en todos los proyectos. Esto ayudará a simplificar la evaluación.
- Dar inicio a los proyectos en las primeras etapas del curso mientras aprenden las bases teóricas de los siguientes proyectos.

Fecha de Entrega: 12/02/2017. Esta fecha es definitiva, no se aceptarán prórrogas ni entregas tarde, bajo ninguna circunstancia.

Método de Entrega: La entrega se realizará mediante el GES en la asignación respectiva. Deben entregarse únicamente los archivos *.java* y **ningún archivo *.class***; deben asegurarse de que todos los archivos compilan correctamente**. Si su código no compila, tendrán una nota automática de 0.** Todos los archivos y carpetas (en caso usen paquetes) debe entregarse dentro de un archivo *.zip* cuyo nombre tenga el formato: <carné>-proy1.zip.

Especificaciones:

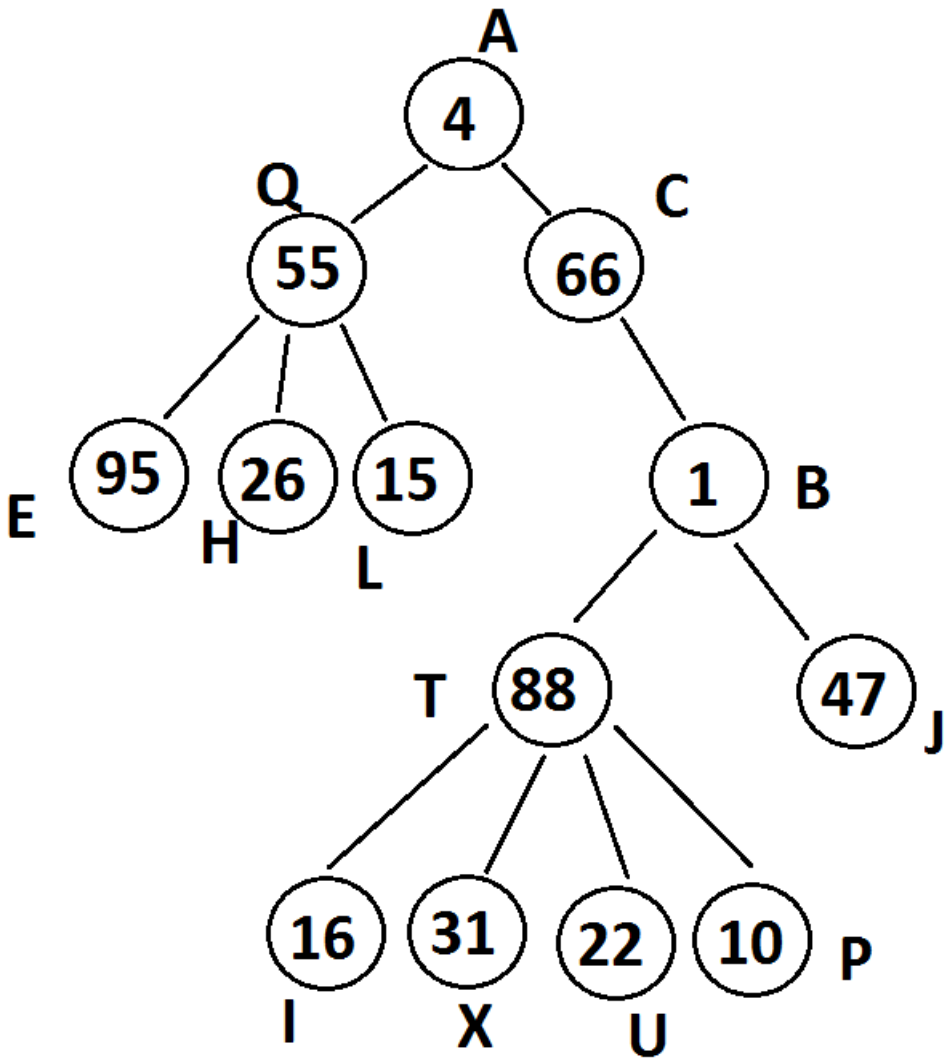
Codigo Base: En el GES, en la sección de **Material de Apoyo** encontraran una carpeta llamada *pj1*. Esta carpeta contiene los archivos base para su proyecto. Deben implementar el método principal en la clase **Rec.java**, y si desean, pueden usar cualquier otra clase auxiliar que necesiten.

Que Deben Hacer: En su programa, deben implementar 3 algoritmos de **forma recursiva**, si no utilizan recursividad, y lo hacen de forma iterativa, tendrán automáticamente un 0 en el punteo que valga ese ejercicio. Los algoritmos que deben implementar son:

- Factorial
- Función de Ackerman
- Búsqueda de un valor en una estructura de árbol

Para el Factorial y la Función de Ackerman, ustedes únicamente deben cumplir con los parámetros de ejecución y la firma de los métodos (ambos descritos más adelante). Para la búsqueda en el árbol, sin embargo, hay algunas especificaciones extra que deben cumplir:

Dentro de la carpeta pj1 se encuentran los archivos TreeNode.java y TreeGenerator.java. TreeNode.java es la clase que implementa un árbol. En este árbol cada nodo contiene un dato y puede contener otros N nodos. Lean los métodos y entiéndanlos, les servirán para que creen sus propios árboles y puedan hacer pruebas con su algoritmo de búsqueda. De esta clase, el único método que deben implementar ustedes es **searchPath(int v)**, en donde deben buscar el valor de v en el árbol, de forma recursiva, y devolver el camino que hay que seguir desde la raíz hasta el nodo que contenga el valor. Ejemplo:



Si en el siguiente árbol, buscamos el número 31, su método **searchPath(int v)** debe devolver algo como esto: "[A]->[C]->[B]->[T]->[X]" Noten que los nombres de los nodos (las letras) no se repiten, ni tampoco los valores que contienen. También tomen en cuenta que los arboles no necesariamente estarán ordenados de ninguna forma.

Es buena idea que prueben su algoritmo creando ustedes sus propios árboles, pero como el objetivo del proyecto no es la implementación de estructuras de datos, el archivo TreeGenerator.java tiene 5 ***variables de clase (static).*** Estas variables llamadas T1, T2, T3, T4 y T5 son arboles ya generados, creados con los métodos tree1(), tree2() …. tree5(), que se encuentran más abajo en el mismo archivo. Pueden usarlas para hacer sus pruebas y son también las que usaremos para ejecutar la búsqueda desde línea de comandos.

Acerca de los parámetros de ejecución:

Una vez terminado su programa, se debe ejecutar de la siguiente forma:

java Rec (-d|-r) (fact|ack|search) (extra_params)

Ejemplos:

//>java Rec -r fact 3	//Calculara 3! desplegando únicamente el resultado
//>java Rec -d fact 3	//Calculara 3! desplegando toda la secuencia del cálculo hasta llegar al resultado final.
//>java Rec -r ack 2 3	//Calculara Ackerman(2,3) y desplegara únicamente el resultado
//>java Rec -d ack 2 3	//Calculara Ackerman(2,3) y deplegara toda la secuencia del calculo hasta llegar al resultado final
//>java Rec -r search T1 2	//Buscara en el arbol TreeGenerator.T1 el valor 2, desplegara "SI" si el valor está en el árbol y "NO" si no esta
//> java Rec -d search T3 10	//Buscara en el arbol TreeGenerator.T3 el valor 10, y desplegara el camino que hay que tomar desde la raíz hasta el nodo si lo encuentra. Desplegara "NO" si no lo encuentra

NOTA: en -search, el primer parámetro solo puede ser T1, T2, T3, T4 o T5 para ser válido, porque TreeGenerator.java solo tiene 5 árboles.

Acerca del nombre de los métodos:

En su código, en alguna de las clases que implementen (puede ser incluso en Rec.java) deberá incluir los siguientes métodos restringiendo su nombre y tipo a:

Factorial de un número: **int fact(int num)**

Ackerman(x,y): **int ack(int a, int b)**

Búsqueda de un valor : **String searchPath(int num)**

NOTA : Las rutinas deben realizar UNICAMENTE utilizando funciones recursivas, es decir, no pueden realizarse la iteraciones necesarias mediante ciclos (while, for, etc.), la utilización de estos y no de recursividad anulará la posible nota de dicha rutina.

Otras especificaciones:

El programa deberá verificar para cada rutina recursiva que la cantidad y el tipo de parámetros además de su ordenamiento sean según lo especificado anteriormente. Por ejemplo, al momento de ejecutar el programa, los errores podrían ser:

//> java Rec fact 11x	El parámetro 11x no es de tipo entero
//> java Rec ack 11	Hace falta un parámetro
//> java Rec ack 11 2 –d	El parámetro –d no se encuentre en la posición requerida (posición cero del arreglo)
//> java Rec fib arl	El parámetro arl no es de tipo entero

Su programa debe estar diseñado para detectar cualquier error que puedan pensar, no únicamente los errores especificados arriba.

COPIA:

Los casos de copia serán sancionados según el reglamento de la Universidad Galileo y con una nota de -100 en la actividad. Tomen en cuenta que usaremos MOSS para verificar copias.