UNIVERSIDAD DE CARABOBO
Facultad Experimental de Ciencias y Tecnología
Departamento de Computación
U.A. de Algoritmos, Programación y Lenguajes
CAO403: Programación II

PROYECTO
Período Lectivo 1-2016
(Valor: 10%)

1. INSTRUCCIONES

- Este proyecto de programación es una evaluación estrictamente en pareja y del mismo grupo de laboratorio; por lo tanto, durante la realización del mismo, no está permitido prestar o recibir ayuda de otro(s) estudiante(s) que no formen parte del equipo. ESTÁ permitido el uso de códigos fuentes en digital hechos por usted para el desarrollo del proyecto.
- El proyecto tiene una duración estimada de once (11) días contados a partir del lunes **4 de julio de 2016** al viernes **15 de julio de 2016**.
- La entrega del proyecto se realizara por el entorno virtual hasta el viernes **15 de julio de 2016** hasta las **11:59 pm.**
- El estudiante que infrinja alguna de las instrucciones anteriores será retirado del proyecto y se le considerará aplazado con la nota mínima (Artículo 34 de las Normas de Evaluación de los Aprendizajes).
- El ejercicio propuesto requiere la lectura de datos de la entrada estándar (*standard input*) y la escritura de resultados en la salida estándar (*standard output*).
- Debe proveer un archivo *makefile* que permita compilar su proyecto, la falta del mismo conllevará a la no revisión de su proyecto, adicionalmente dicho *makefile* debe generar un ejecutable con el nombre de "*quadtree*", con el fin de facilitar las labores de corrección de su proyecto, el no hacerlo generara puntos menos en su nota final del proyecto.

2. ACTIVIDADES A REALIZAR

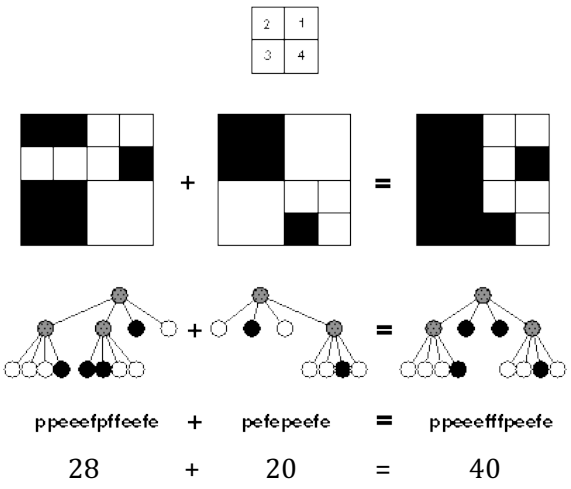
Planteamiento del Problema

Un *quadtree* es un formato utilizado para la representación de imágenes. La idea fundamental tras el *quadtree* es que cualquier imagen puede ser dividida en cuatro cuadrantes. Cada cuadrante puede a su vez ser dividido en cuatro sub–cuadrantes, y así sucesivamente. De esta manera, un *quadtree* es un árbol 4–ario que permite representar de manera compacta una imagen, que en otro caso podría ocupar grandes cantidades de memoria. En el *quadtree* la imagen es representada por un nodo padre, mientras que los cuatro cuadrantes son representados por cuatro nodos hijos, en un orden predeterminado.

Si todos los *pixels* de la imagen son de un único color, ésta puede ser representada por un *quadtree* de un solo nodo. En general, un cuadrante sólo requiere ser subdividido si contiene *pixels* de diferentes colores. En consecuencia, un *quadtree* no es necesariamente un árbol balanceado.

Suponga que usted trabaja con imágenes en blanco y negro de 8 x 8 unidades, para un total de 64 *pixels* por imagen. Una de las operaciones que se efectúan sobre este tipo de objetos es la suma de dos imágenes para formar una nueva. En la imagen resultante, un *píxel* es negro si es negro en alguna de las imágenes componentes. En otro caso el *píxel* es blanco. Otra operación de interés consiste en contar el número de *pixels* en negro que componen una imagen dada.

En la siguiente figura se presenta un ejemplo de suma de imágenes. Para cada imagen se muestra su *quadtree* asociado, la secuencia en preorden que lo representa para efectos del archivo de entrada (esto se definirá más adelante) y el número de *pixels* en negro que contiene. El orden de numeración de los cuadrantes se muestra al inicio del gráfico.



De acuerdo a lo expuesto anteriormente, se puede definir la CLASS *Quadtree* de la siguiente manera:

Class Quadtree[Color]

SINTAXIS

$nulo_{qt}$:		→	Quadtree
es_nulo_{qt} :	Quadtree	→	Booleano
$crear_{qt}$:	Lista[Color]	→	Quadtree
$color_{qt}$:	Quadtree	→	Color
$union_{qt}$:	Quadtree × Quadtree	→	Quadtree
$pixels\_en\_negro_{qt}$:	Quadtree × Entero	→	Entero

SEMÁNTICA

{ Pre: }
función $nulo_{qt}$ ( ) : Quadtree;
{ Post: Retorna un *Quadtree* nulo }

{ Pre: }
función es_nulo_{qt} (Qt : Quadtree) : Booleano;
{ Post: Indica si Qt es nulo }

{ Pre: $\neg es\_vacia_L(lista\_preorden)$ }
función $crear_{qt}$ (lista_preorden : Lista[Color]) : Quadtree;
{ Post: Crea un *Quadtree* a partir de la lista que contiene el color de sus nodos en preorden}

{ Pre: $\neg es\_nulo_{qt}(Qt)$}
función $color_{qt}$ (Qt: Quadtree) : Color;
{ Post: Retorna el color del nodo raíz de Qt }

{ Pre: $\neg es\_nulo_{qt}(Qt_1) \wedge \neg es\_nulo_{qt}(Qt_2)$ }
función $union_{qt}$ (Qt_1, Qt_2 : Quadtree) : Qadtree;
{ Post: Retorna el *quadtree* resultante de la suma de la suma de Qt_1 y Qt_2 }

{ Pre: ¬es_nulo$_{qt}$(Qt) }
función pixels_en_negro$_{qt}$(Qt: Quadtree, max_alt: Entero) : Entero;
{ Post: Retorna la cantidad de *pixels* negros de Qt cuya altura máxima es max_alt }

Para la implementación de la **CLASS Quadtree** considere la representación de árboles con vector de apuntadores vista en clases, además el tipo de dato a usar por el Quadtree es el siguiente:

Type

Color = (BLANCO, NEGRO, GRIS);

Actividades a cumplir

- Implemente cada operación de la **CLASS Quadtree** en términos de las estructuras de datos definidas anteriormente.
- Utilice las operaciones de la **CLASS Lista** que sean necesarias para el manejo de la lista de nodos especificada en la operación crear$_{qt}$.
- Utilice la **CLASS Quadtree** para elaborar una rutina que, dados dos *quadtrees*, genere el *quadtree* resultante de la suma de los mismos y calcule el número de *pixels* negros en la imagen resultante.

Formato de entrada

La primera línea de entrada especifica el número de casos de prueba. La entrada para cada caso de prueba consiste de dos cadenas da caracteres, cada una en su propia línea. Cada cadena es la representación en preorden de un *quadtree*, en la cual la letra ´p´ indica un nodo padre (GRIS), la letra ´f´ (full) indica un cuadrante negro y la letra ´e´ (empty) indica un cuadrante blanco. Se garantiza que cada cadena representa un *quadtree* válido cuya altura es menor o igual a tres (3) debido a las dimensiones dela imagen (8 x 8 *pixels*).

Formato de salida

Para cada caso de prueba debe generarse la línea de texto “Hay X pixels negros”, donde X es el número de *pixels* negros en la imagen resultante.

Ejemplo de entrada

```
3
ppeeefpffeefe
pefepeefe
peeef
peefe
peeef
peepefefe
```

Ejemplo de salida

```
Hay 40 pixels negros.
Hay 32 pixels negros.
Hay 24 pixels negros.
```

3. Observaciones

- No debe mostrar por pantalla ningún tipo de mensaje al usuario para su interacción con el programa, ninguna interfaz gráfica desarrollada por usted, será corregida positivamente o le aportara ningún punto extra para su calificación final
- El formato de salida debe ser respetado como se indica en el enunciado, de lo contrario acarreará con puntos menos en su nota final.