Il faut travailler **sérieusement** le corrigé avec votre copie et ce compte-rendu.

Ayez le troisième œil pour déterminer l'avenir des variables !

Rien à dire sur celui-là, il n'y a eu que quelques distraits pour se tromper.

Max adore la récursivité

1.
2. Ne pas se tromper entre `//` et `%`.
3. Une bonne fois pour toute, il faut savoir que si `M=L[i:j]`, alors `M=[L[i],L[i+1],...,L[j-1]]`. C'est la même chose qu'un `range` dans une boucle `for`, la commande `for k in range(i,j)`, `k` va prendre toutes les valeurs entre `i` **inclus** et `j` exclus (et donc s'arrêter à `j-1`).
4. Commandes déjà utilisées lors du TP sur la récursivité. Il n'y a pas de $+1$ ou de -1 au vu de la remarque précédente.
5. Exercice déjà vu lors du TP récursivité. Je n'ai pas sanctionné pour cette fois-ci le code suivant mais c'est extrêmement maladroit :

```
def Max(L):
        if len(L)==1:
                return L[0]
        else:
                G=L[0:len(L)//2]
                D=L[len(L)//2:len(L)]
                if Max(G)>Max(D):
                        return Max(G)
                else:
                        return Max(D)
```

   En effet, à chaque fois qu'une fonction est appelée, le calcul est refait, ainsi `Max(G)` va être calculé deux fois au lieu qu'une si `Max(G)>Max(D)`, sinon ce sera `Max(D)`. Sauf que nous sommes en récursif, et s'il faut calculer deux fois `Max(G)`, cela veut dire qu'il va falloir calculer quatre fois le maximum de la propre partie gauche de `G` ou quatre fois sa partie droite, et de même en continuant. Ce qui fait que c'est inutilisable en pratique pour des listes de grandes tailles.

6. De même le code suivant est très maladroit :

```
def CombienDeMax(L):
        C=0
        for i in range(len(L)):
                if L[i]==Max(L):
                        C=C+1
        return C
```

   En effet, pour chaque `i`, vous allez calculer `Max(L)`. Ainsi, si `n=len(L)`, vous allez calculer n fois le maximum de la liste. Alors que si en début de programme, vous aviez écrit `M=Max(L)`, ce maximum aurait été calculé une bonne fois pour toute, et vous auriez à chaque fois comparé `L[i]` à `M`.

Soyez iconoclastes avec les images

On note `L=[[1,2,3],[4,5,6]]`.

1. Il faut avoir compris comment fonctionne une liste.
2. Commande déjà vue lors du TP8
3. Commande déjà vue lors du TP8
4. Un peu de dénombrement. Une couleur est rien de plus représenté comme un élément de $[\![\,0\,;255\,]\!]^3$. Or cet ensemble à 256^3 éléments.

5. Commande déjà vue lors du TP8
6. Cette question n'était pas si facile, certains ont pris une variable qui va servir de compteur et en faisant une boucle sur les indices de lignes et de colonnes on fait l'instruction `C=C+1` si `Img[i][j]!=Img[0][0]` (pour certains) ou si `Img[i][j]!=Img[i+1][j+1]` (pour d'autres). Dans le premier cas, cela a compté le nombre de pixels différent du pixel en haut à gauche (mais cela ne compte pas le nombre de couleurs différentes), dans le second cas, cela a compté le nombre de pixels dont la couleur est différente de leur voisin décalé d'un cran en bas à droite (mais cela ne compte pas le nombre de couleurs différentes).
7. Attention, on ne vous a pas demandé de modifier la liste `Img`, donc si vous écrivez dans une fonction `Img[i][j]=...` vous modifiez `Img` en dehors de la fonction par ce qu'on appelle un **effet de bord**. Il fallait procéder comme dans le TP image, partir d'une autre liste de listes initialement remplie de 0 et lui affecter une valeur.
8. Même remarque qu'à la question précédente.
9. Même remarque qu'à la question précédente sauf qu'il fallait définir une image `T` avec les bonnes tailles. Il fallait faire un dessin pour se convaincre que

$$T[i][j] = \frac{T[2i][2j] + T[2i+1][2j] + T[2i][2j+1] + T[2i+1][2j+1]}{4}$$

   Attention, en Python il faut écrire `2*i` et non `2i`.
10. De même, il faut définir une image, initialement nulle, de la bonne taille et faire un dessin pour comprendre que :

$$T[i][j] = \frac{1}{k^2} \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} T[k*i+a][k*j+b]$$

11. Il fallait comprendre que diviser la taille d'une image par 2^k c'est d'abord diviser sa taille par 2 puis diviser la taille de l'image obtenue par 2^{k-1} (ou faire les choses en sens inverse).
12. Comme en maths, une matrice et sa transposée n'ont pas le même nombre de lignes et de colonnes [1] Il fallait adapter le nombre de lignes et de colonnes ainsi les boucles `for`, sinon c'est des **out of range** assurés.

1. En quelque sorte les tailles se transposent également.