Devoir surveillé n°6 – NSI
Correction

Exercice 1

1. Puisqu'il faut dépasser la moitié, les effectifs possibles d'un élément absolument majoritaire dans une liste de taille 10 sont les nombres entiers de 6 à 10.

Partie A : Calcul des effectifs de chaque élément sans dictionnaire

2. Voici le code :

```
def effectif(val, lst):
    total = 0
    for x in lst:
        if x == val:
            total += 1
    return total
```

3. La fonction `effectif` parcourt chaque élément de la liste et le compare au nombre `val`. Il n'y a pas d'autre comparaison dans la fonction.

   Dans la liste `[1, 4, 1, 6, 1, 7, 2, 1, 1]` il y a 9 éléments, donc le nombre de comparaisons est 9.

4. On peut avoir le code suivant :

```
def majo_abs1(lst):
    moitie = len(lst) // 2
    for elt in lst:
        if effectif(elt, lst) > moitie:
            return elt
    return None
```

5. L'élément majoritaire de la liste `[1, 4, 1, 6, 1, 7, 2, 1, 1]` est `1`, qui est le premier élément de la liste.

   Lors du parcourt, la première valeur de `elt` est donc `1`.

   Lors de l'exécution de `effectif(elt, lst)` on a vu qu'il y a 9 comparaisons.

   De plus, il y a une comparaison entre ce nombre et la variable `moitie`.

   Comme la condition `effectif(elt, lst) > moitie`, la fonction retourne tout de suite un résultat, il n'y a donc pas d'autre comparaison.

   Au total, il y a donc eu $9 + 1 = 10$ comparaisons.

Partie B : Calcul des effectifs de chaque élément dans un dictionnaire

6. Voici le code complété :

```
1  def eff_dico(lst):
2      dico_sortie = {}
3      for x in lst :
4          if x in dico_sortie:
5              dico_sortie[x] += 1
6          else:
7              dico_sortie[x] = 1
8      return dico_sortie
```

7. Voici un code possible :

```python
def majo_abs2(lst):
    moitie = len(lst) // 2
    dico_effectifs = eff_dico(lst)
    for x,n in dico_effectifs.items():
        if n > moitie:
            return n
    return None
```

Partie C : par la méthode « diviser pour régner »

8. si `n = 1`, l'élément absolument majoritaire de `lst` est l'unique élément de cette liste, autrement dit `lst[0]`.
9. Si, ni `lst1` ni `lst2` n'admet d'élément absolument majoritaire, `lst` n'admet pas d'élément absolument majoritaire car :

   Dans le meilleur des cas, l'élément qui apparaît le plus dans chacune des deux listes `lst1` et `lst2` apparaît `(n//2)//2`, soit `n//4` fois (la moitié de leur effectif), et si, toujours dans le meilleur cas, il s'agit de la même valeur, alors dans `lst` il apparaît donc `2*(n//2)//2` fois, soit `n//2)` fois, ce qui n'est pas majoritaire.

   Autre raisonnement : si on suppose qu'il existe un élément majoritaire dans `lst`, alors celui-ci apparaît un nombre `m` de fois strictement supérieur à `n//2`.

   Cet élément apparaît alors `m1` fois dans `lst1` et `m2` fois dans `lst2`, avec `m1+m2=m`.

   Mais alors, nécessairement, `m1` ou `m2` est strictement supérieur à `n//4` (si les deux étaient strictement inférieurs, alors leur somme resterait inférieure ou égale à `n//2`, or on a supposé qu'elle était strictement supérieure). Autrement dit, l'élément serait alors majoritaire soit dans `lst1`, soit dans `lst2`, ce qui est contradictoire.
10. Il suffit de vérifier que l'effectif de `maj1` dans `lst` est strictement supérieur à `n//2`.
11. Voici le code complété :

```python
def majo_abs3(lst):
    n = len(lst)
    if n == 1:
        return lst[0] # cas de base
    else:
        lst_g = lst[:n//2]
        lst_d = lst[n//2:]
        maj_g = majo_abs3(lst_g)
        maj_d = majo_abs3(lst_d)
        if maj_g is not None:
            eff = effectif(maj_g,lst)
            if eff > n/2:
                return maj_g
        if maj_d is not None:
            eff = effectif(maj_d,lst)
            if eff > n/2:
                return maj_d
```