Exercice 3

1a

```
def total_hors_reduction(tab):
    total = 0
    for pa in tab:
        total = total +  pa
    return total
```

on acceptera aussi un simple :

```
def total_hors_reduction(tab):
    return sum(tab)
```

1b

```
def offre_bienvenue(tab):
    somme = 0
    longueur=len(tab)
    if longueur > 0:
        somme = tab[0]*0.8
    if longueur > 1:
        somme = somme + tab[1]*0.7
    if longueur > 2:
        for i in range(2,longueur):
            somme=somme+tab[i]
    return somme
```

2

```
def prix_solde(tab):
    longueur = len(tab)
    reduc = 1
    if longueur == 1 :
        reduc = 0.9
    if longueur == 2 :
        reduc = 0.8
    if longueur == 3 :
        reduc = 0.7
    if longueur == 4 :
        reduc = 0.6
    if longueur >= 5 :
        reduc = 0.5
    return reduc*total_hors_reduction(tab)
```

autre possibilité plus “courte” :

```
def prix_soldeb(tab):
    return total_hors_reduction(tab)*(1-0.1*min(5,len(tab)))
```

3a
Dans cette question nous partons du principe que tab n'est pas vide.

```
def minimum(tab):
    mini = tab[0]
    for pa in tab:
        if pa <  mini:
            mini = pa
    return mini
```

on acceptera aussi :

```
def minimum(tab):
    return min(tab)
```

3b

```
def offre_bon_client(tab):
    longueur = len(tab)
    total = total_hors_reduction(tab)
    if longueur >= 2:
        total = total - minimum(tab)
    return total
```

4a

plusieurs solutions possibles
[30.5, 20.0, 35.0, 15.0, 6.0, 5.0, 10.5] => total après promotion = 122 - 20 - 5 = 97
4b

[35, 30.5, 20.0, 15.0, 10.5, 6.0, 5.0] => total après promotion = 122 - 20 - 6 = 96

4c

Pour avoir le prix après promotion de déstockage le plus bas possible, il faut trier le tableau dans l'ordre décroissant. On peut donc utiliser un algorithme de tri (tri par sélection, tri par insertion ou tri fusion)