1 Recherche dichotomique dans une liste triée

Principe:

On utilise un intervalle de recherche défini par les indices `[iA, iB]` (indices inclus). A chaque itération, on prend l'indice `iC` du milieu de l'intervalle de recherche et on lit la valeur de l'élément `L[iC]` qui correspond :

- soit x est trouvé, on s'arrête en renvoyant True,
- soit x est **supérieure à** la valeur du point milieu. Cela signifie qu'il faut poursuivre la recherche dans la moitié supérieure de la liste triée (c'est-à-dire dans l'intervalle `[iC+1, iB]`), car les valeurs plus grandes s'y trouvent. On poursuit donc la recherche en actualisant la borne inférieure par l'instruction `iA = iC + 1`.
- soit x est **inférieure à** la valeur du point milieu. On poursuit la recherche en actualisant la borne supérieure par l'instruction `iB = iC - 1` afin que poursuivre la recherche dans l'intervalle `[iA, iC-1]`

L'algorithme se poursuit donc tant que l'intervalle de recherche est au moins de largeur 1.

Compléter le code ci-dessous

[ ]:
```
def rechercheDico(L,x) : # L est supposée triée par ordre croissant
    iA, iB = 0, len(L)-1
    while iB  ... :    # tant que l'intervalle de recherche contient contient au␣
 ↪moins un élément
        iC = ...  # indice du point milieu arrondi à la valeur entière
        print(iA,iB,iC,L[iC]) # affichage pour le suivi du déroulement de␣
 ↪l'algorithme
        # 3 possibilités
        if L[iC] ... :  # trouvé !
            ...
        if L[iC] < x :
            iA = ...
        else :
            iB = ...
    return False
```

[13]:
```
# code de validation à exécuter
import numpy.random as rd
a, b, N = -5000, 25000, 5000
L = rd.randint(a, b, N)
print("Liste = ",L)
L.sort()
print("Liste = ",L)
print("Cet élément est dans la liste : ", L[N//3])
```

```
Liste =  [ 8025 16848 18065 ...  3865 22125 21317]
Liste =  [-5000 -4991 -4975 ... 24993 24997 24999]
Cet élément est dans la liste :  4774
```

[ ]:
```
rechercheDico(L,...) # à compléter
```

[ ]:
```

```