Ce document recense des fonctions du cours et des TP que vous devez savoir écrire rapidement et sans erreur. Lorsque plusieurs versions d'une même fonction sont données, toutes les versions sont à connaître. Pensez à revoir également les fiches de révision des DS précédents.

1 Tris

⋆ Tri par insertion :

```
# Hypothèse: T est triée
def insertion(T, x):
    i = 0
    while i < len(T) and T[i] < x:
        i += 1
    return T[:i] + [x] + T[i:]
```

```
def tri_insertion(L):
    T = []
    for e in L:
        T = insertion(T, e)
    return T
```

⋆ Tri par sélection.

```
# Hypothèse: L est non vide
def mini(L):
    i_min = 0
    for i in range(1,len(L)):
        if L[i] < L[i_min]:
            i_min = i
    return i_min
```

```
def tri_selection(L):
    T = []
    for _ in range(len(L)):
        i = mini(L)
        T.append(L[i])
        L = L[:i] + L[i+1:]
    return T
```

⋆ Tri par comptage. On suppose que chaque élément `e` de `L` vérifie $0 \leqslant$ `e` $\leqslant$ `e_max`.

```
def comptage(L, e_max):
    M = [0 for _ in range(e_max + 1)]
    for e in L:
        M[e] += 1
    return M
```

```
def tri_comptage(L, e_max):
    T = []
    M = comptage(L, e_max)
    for i in range(len(M)):
        for _ in range(M[i]):
            T.append(i)
    return T
```

2 Représentation en base 2

⋆ Renvoie la représentation binaire de `n`.

```
# Hypothèse n >= 0
# Renvoie "" pour n = 0
def int_to_bin(n):
    s = ""
    while n > 0:
        if n % 2 == 0:
            s = "0" + s
        else:
            s = "1" + s
        n = n//2
    return s
```

⋆ Calcule un entier à partir de sa représentation binaire.

```
# Solution 1
# Hypothèse: s ne contient que des 0 et des 1
def bin_to_int(s):
    n = 0
    for i in range(len(s)):
        if s[i] == "1":
            n = n + 2**(len(s)-1-i)
    return n
```

```
# Solution 2 (évalutation de Horner)
# Hypothèse: s ne contient que des 0 et des 1
def bin_to_int(s):
    n = 0
    for i in range(len(s)):
        if s[i] == "0":
            n = 2*n
        else:
            n = 2*n + 1
    return n
```