On définit les deux opérateurs suivants pour un tableau $A = [a_0, a_1, \ldots, a_{n-1}]$ de n entiers :

- PRESCAN(A) renvoie le tableau $[0, a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots, a_0 + a_1 + \ldots + a_{n-2}]$
- SCAN(A) renvoie le tableau $[a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots, a_0 + a_1 + \ldots + a_{n-1}]$

Nous avons vu section 1.2 comment réaliser ces deux opérateurs en temps $O(\log n)$ sur une P-RAM EREW.

▶ **Question 1.** Étant donné un tableau de booléens *Flags*, que fait la procédure suivante ?

```
SPLIT(A,Flags)
  Idown ←PRESCAN(not(Flags))
  Iup ←n - REVERSE(SCAN(REVERSE(Flags)))
  Pour i = 1 à n en parallèle :
     Si Flags(i) Alors
        Index[i] ←Iup[i]
     Sinon
        Index[i] ←Idown[i]
  Result ←PERMUTE(A,Index)
  Renvoyer Result
```

Les noms des différentes fonctions sont relativement intuitifs ; en particulier, REVERSE renverse le tableau, et PERMUTE(A,*Index*) réordonne le tableau A selon la permutation *Index*. L'horrible expression REVERSE(SCAN(REVERSE(*Flags*))) effectue simplement un SCAN à partir de la fin du tableau *Flags*, dont les éléments sont considérés comme des entiers.

Voici un exemple d'utilisation :

A	=	[	5	7	3	1	4	2	7	2	]
Flags	=	[	1	1	1	1	0	0	1	0	]
Idown	=	[	0	0	0	0	$\boxed{0}$	$\boxed{1}$	2	$\boxed{2}$	]
Iup	=	[	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{6}$	7	7	$\boxed{7}$	8	]
Index	=	[	3	4	5	6	0	1	7	2	]
Result	=	[	4	2	2	5	7	3	1	7	]

Quel est le coût de la fonction SPLIT ?

▶ **Question 2.** On considère la procédure MYSTÈRE suivante :

```
MYSTÈRE(A, Number_Of_Bits)
  Pour i = 0 à Number_Of_bits - 1 :
     bit(i) ←tableau indiquant si le i-ème bit des éléments de A est à 1
     A ←SPLIT(A,bit(i))
```

1) Faire tourner la procédure sur $A = [5, 7, 3, 1, 4, 2, 7, 2]$ avec $Number\_Of\_Bits = 3$.

2) Que fait la procédure MYSTÈRE ?

3) Avec des entrées de taille $O(\log n)$ bits, quelle est la complexité avec n processeurs ? Et avec seulement p processeurs ? Quelles sont les valeurs de p les plus intéressantes ?