Exercice 5 - initiation à l'encapsulation

Nous allons créer un ensemble de sous programmes permettant de manipuler des tableaux triés. Ce tableaux sont de taille N, vous prendrez le soin de vérifier qu'il ne se produit aucun débordement d'indice. Nous voulons pouvoir faire varier le nombre d'élements placés dans le tableau, c'est-à-dire en ajouter et en enlever. Comme les tableaux sont de taille fixe, il convient de placer dans une variable le nombre d'éléments significatifs du tableau. On utilisera pour ce faire la première case du tableau, c'est-à-dire celle d'indice 1. Par exemple, le tableau

0	5	3	6	2	4	5	8	3	6	2	7	3	4	0

ne contient aucun élément, le 0 qui se trouve au début du tableau nous indique qu'aucun élément n'est significatif. Les éléments significatifs sont placés juste après le premier élément. Donc, s'il y a 7 éléments significatifs dans le tableau T, on aura $T(1) = 7$, et ces 7 éléments seront stockés dans $T(2), \ldots, T(8)$. Par exemple, le tableau

3	2	3	6	2	4	5	8	3	6	2	7	3	4	0

contient les 3 éléments 2, 3 et 6. Le 3 se trouvant dans le premier emplacement du tableau nous indique que seuls les 3 premiers éléments sont significatifs, les autres sont ignorés. Notez que nous ferons en sorte que les éléments **significatifs** soient disposés par ordre croissant.

1. Ecrire la procédure `affiche(numérique : T[N])`. Ce sous programme affiche tous les éléments **significatifs** du tableau T.
2. Ecrire la fonction `taille(numérique : T[N]) : numérique`. Ce sous programme retourne le nombre d'éléments significatifs se trouvant dans le tableau T. **Ne cherchez pas quelquechose de compliqué, il y a une seule instruction dans ce sous-programme !**
3. Ecrire la fonction `estVide(numérique : T[N]) : booléen`. Ce sous programme retourne vrai si et seulement si le tableau T est vide.
4. Ecrire la fonction `estPlein(numériques : T[N]) : booléen`. Ce sous programme retourne vrai si et seulement si le tableau T est plein.
5. Ecrire la procédure `initialise(numériques : T[N], k)`. Ce sous programme initialise T de la sorte : | k | 1 | 2 | ... | k | ... | on supposera que k est toujours strictement inférieur à N.
6. Ecrire la fonction `indice(numériques : T[N] e/s, x) : numérique`. Ce sous-programme retourne l'indice de l'élément x dans le tableau T, -1 si x ne se trouve pas dans T. **On part du principe que le tableau T est trié**, vous devez en tenir compte dans votre sous-programme.
7. Ecrire la procédure `supprimeFin(numériques : T[N] e/s)`. Ce sous programme supprime le dernier élément du tableau en le rendant non significatif. Si le tableau est vide, ce sous-programme ne fait rien.
8. Ecrire la procédure `ajouteFin(numériques : T[N] e/s, x)`. Ce sous-programme ajoute l'élément x à la fin du tableau T, en vérifiant que le tableau reste trié, et qu'aucun débordement ne se produit. Si l'ajout est impossible, ce sous-programme ne fait rien.

9. Ecrire la procédure `decalageGauche(numérique : T[N] e/s, i, j)`. Ce sous programme décale vers la gauche la tranche $T(i), \ldots, T(j)$. Vous partirez du principe que les valeurs de i et j sont telles qu'aucun débordement d'indice ne peut se produire. Vous ne modifierez pas la valeur de $T(1)$. Si cette procédure est appelée sur le tableau

3	1	4	6	9	12	16	21	120

avec $i = 4$ et $j = 7$, on obtient

3	1	6	9	12	16	16	21	120

Notez bien que ce n'est pas une permutation circulaire.

10. Ecrire la procédure `decalageDroite(numériques : T[N] e/s, i, j)`. Ce sous programme décale vers la droite la tranche $T(i), \ldots, T(j)$. Vous partirez du principe que les valeurs de i et j sont telles qu'aucun débordement d'indice ne peut se produire. Vous ne modifierez pas la valeur de $T(1)$.

11. Ecrire la procédure `insere(numériques : T[N] e/s, x)`. Ce sous programme insère l'élément x dans le tableau en **veillant à ce que celui-ci soit trié après l'insertion**. Si le tableau est plein, ou que x est déjà dans le tableau, alors ce sous-programme ne fait rien.

12. Ecrire la procédure `supprime(numériques : T[N] e/s, x)`. Ce sous programme supprime l'élément x du tableau . Si x ne se trouve pas dans T, alors ce sous-programme ne fait rien.

13. Nous souhaitons perfectionner la fonction `indice`. Au lieu de parcourir tous les éléments significatifs du tableau, nous allons les séparer en deux tranches et vérifier dans quelle tranche peut se trouver l'élément x que l'on recherche. Par exemple, si l'on cherche l'élément 4 dans le tableau

9	1	4	6	9	12	14	16	21	120

on commence par l'élément se trouvant au milieu du tableau, à savoir 12. On sépare donc les éléments significatifs en deux tranches, à savoir

1	4	6	9

et

14	16	21	120

Comme $12 \neq 4$, il est nécessaire de poursuivre la recherche. En comparant 12 à 4, on déduit, parce que le tableau est trié, que l'élément 4 ne peut se trouver que dans la tranche

1	4	6	9

des 4 premiers éléments significatifs du tableau. On applique le même procédé à cette tranche. On peut couper cette tranche en deux soit de part et d'autre du 4, soit de part et d'autre du 6. Si nous choisissons 6, il reste deux tranches :

1	4

et

9

on poursuit la recherche jusqu'à ce qu'il reste un seul ou aucun élément dans la tranche considérée. Réécrire la fonction `indice` en utilisant la méthode décrite précédement.