ITI3 3 Novembre 2025

Algorithmique avancée et programmation C

Durée : *1h30*
Documents autorisés : **AUCUN**

Remarques :
— Vos réponses aux exercices 2 et 3 seront sur des copies doubles différentes
— Veuillez lire attentivement les questions avant de répondre ;
— Rendez une copie propre ;
— N'utilisez pas de crayon à papier sur votre copie ;
— Pour chaque exercice, il est indiqué, entre parenthèses, le nombre d'Attendus d'Apprentissage Disciplinaires (AAD) évalués ;
— Vous trouverez en annexe les signatures de fonctions et procédures sur des collections, SDD nécessaires à certains exercices.

1 QCM (3 AAD)

Répondez au qcm ci joint (à rendre avec votre copie). La note de chaque question peut varier de -1 à $+1$.

Soit B le nombre de bonnes réponses à une question et soit M le nombre de mauvaises réponses à cette même question ($B+M$ est égal au nombre de réponses à la question). Chaque bonne réponse cochée rapporte $1/B$ points et chaque mauvaise réponse $-1/M$ points.

2 L'indice de la dernière occurrence d'un entier (8 AAD)

Écrire une fonction récursive, `indiceMax`, qui détermine, en $O(log_2(n))$, le plus grand indice d'un entier e, obligatoirement présent dans un tableau t de nb entiers, trié dans l'ordre croissant. Cet entier peut bien entendu être présent plusieurs fois dans le tableau t.

3 Multiensemble (10 AAD)

D'après Wikipédia, un « multiensemble [...] est une sorte d'ensemble dans lequel chaque élément peut apparaître plusieurs fois. [...] On nomme multiplicité d'un élément donné le nombre de fois où il apparaît.

Formellement, un multiensemble est un couple (A, m) où A est un ensemble appelé support et m une fonction de A dans l'ensemble des entiers naturels, appelée multiplicité. Dans le multiensemble (A, m), l'élément x apparaît $m(x)$ fois. »

3.1 Analyse

Soit le TAD `MultiEnsemble` possédant les opérations suivantes :
— obtenir un multiensemble vide ;
— savoir si un multiensemble est vide ;
— ajouter un élément ;
— savoir si un élément est présent ;
— obtenir la multiplicité d'un élément (la multiplicité d'un élément non présent est de 0) ;
— supprimer un élément ayant une multiplicité strictement positive. Cette opération décrémente la multiplicité de l'élément. L'élément n'est alors plus considéré comme présent si sa multiplicité est de 0 ;
— obtenir le support.

Formaliser (axiomes compris) le TAD `MultiEnsemble`.

3.2 Conception préliminaire

Donnez les signatures des fonctions et procédures du type `MultiEnsemble`.

3.3 Conception détaillée

Proposez une conception détaillée pour le type `MultiEnsemble` (uniquement le type, on ne vous demande pas les algorithmes de ses opérations).

3.4 Utilisation : l'intersection

D'après Wikipédia anglais, l'intersection de deux multiensembles A et B est un multiensemble C tel que le support de C est l'intersection des supports de A et de B et tel que la multiplicité de ses éléments est le *min* des multiplicités de ces éléments dans A et B.

1. Proposez l'analyse descendante de l'intersection de deux multiensembles en faisant apparaître toutes les opérations que vous utilisez (du TAD `MultiEnsemble` ou autre) en respectant le nommage de Wikipédia.
2. Donnez l'algorithme correspondant.

Annexe

Pour rappel le TAD Ensemble vu en cours est :

Nom: Ensemble

Paramètre: Element

Utilise: **Booleen**,**Naturel**

Opérations:

ensemble:	$\rightarrow$ Ensemble
ajouter:	Ensemble $\times$ Element $\rightarrow$ Ensemble
retirer:	Ensemble $\times$ Element $\rightarrow$ Ensemble
estPresent:	Ensemble $\times$ Element $\rightarrow$ **Booleen**
cardinalite:	Ensemble $\rightarrow$ **Naturel**
union:	Ensemble $\times$ Ensemble $\rightarrow$ Ensemble
intersection:	Ensemble $\times$ Ensemble $\rightarrow$ Ensemble
soustraction:	Ensemble $\times$ Ensemble $\rightarrow$ Ensemble

Axiomes:

- ajouter(ajouter(s,e),e)=ajouter(s,e)
- retirer(ajouter(s,e),e)=s
- estPresent(ajouter(s,e),e)
- non estPresent(retirer(s,e),e)
- cardinalite(ensemble())=0
- cardinalite(ajouter(s,e))=1+cardinalite(s) et non estPresent(s,e)
- cardinalite(ajouter(s,e))=cardinalite(s) et estPresent(s,e)

...