Examen -Décembre 2010

Durée : 2h

Les notes de cours, de travaux dirigés et de travaux pratiques sont autorisées. Le sujet comporte deux pages et deux parties (première partie : 14 points, deuxième partie : 6 points). Vous vous attacherez à soigner la présentation générale et l'orthographe, et vous serez notamment vigilant quant à la syntaxe lors de l'écriture de fragment de code.

Le sujet vous impose de programmer en utilisant des classes et des méthodes. Vous devez respecter cette contrainte !

1 Représentation des multi-ensembles

On considère le codage ASCII des caractères usuels. Ceux-ci sont représentés par un code compris entre 0 et 255. Par exemple, le caractère 'c' est représenté par le code 99 et le caractère '4' par le code 52.

Dans ce problème, on s'intéresse au codage des multi-ensembles construits à partir de l'ensemble $\mathcal{A}$ des codes ASCII des caractères. Un multi-ensemble peut être vu comme un ensemble d'éléments de $\mathcal{A}$ où un élément peut apparaître plusieurs fois. Par exemple, {a,b,d,a} est un multi-ensemble contenant deux occurences de a, une seule de b et une seule de d.

On souhaite utiliser un tableau de taille 256, contenant des valeurs entières pour représenter le nombre d'occurences de chaque caractère de $\mathcal{A}$ dans un multi-ensemble donné.

Question 1 Comment représenteriez-vous le multi-ensemble vide avec cette structure de données ? Le multi-ensemble {a,c,a,d,f,f,f} ? Le multi-ensemble contenant tous les éléments de $\mathcal{A}$ deux fois ?

On considère maintenant la déclaration de classe suivante :

```
#define SIZE 256

class multiset

{
  int t[SIZE];

public:
  multiset(){for (int i=0;i<SIZE;i++) t[i]=0;}
  void add(char c);         // ajoute une occurrence de c
  void remove(char c);      // supprime une occurrence de c
  int nb_occ(char c);       // retourne le nombre d'occurences de c
  bool appartient(char c);  // teste si c appartient au multi-ensemble

  friend ostream& operator<<(ostream &o, multiset m);

};
```

Question 2 Expliquez lequel des trois multi-ensembles de la question précédente le constructeur `multiset` produit. Programmez les méthodes `add`, `remove`, `nb_occ` et `appartient`.

Question 3 Programmez la fonction amie d'affichage d'un multi-ensemble. Expliquez pourquoi l'argument de type `multiset` est passé par valeur. Est-il nécessaire de déclarer cette fonction d'affichage comme une fonction amie de la classe ? Justifiez votre réponse.

On souhaite maintenant étendre la classe avec de nouvelles méthodes. On rajoute donc dans la déclaration de la classe les entêtes suivantes :

```
bool inclus(multiset m); // teste l'inclusion d'un multi-ensemble dans un autre
bool egal(multiset m); // teste l'égalité de deux multi-ensembles
int cardinal_diff();  / compte le nombre d'éléments distincts du multi-ensemble
int cardinal_total(); // compte le nombre total d'éléments du multi-ensemble
```

Question 4 Programmez la méthode `inclus`.
Question 5 Proposez une implantation du test d'égalité entre deux multi-ensembles. On pourra réutiliser le test d'inclusion de la question précédente si nécessaire. Précisez comment surcharger l'opérateur `==` afin de pouvoir écrire directement `n==m` pour tester l'égalité entre deux multi-ensembles n et m.
Question 6 Programmez les fonctions `cardinal_diff` et `cardinal_total`.
Question 7 On souhaite maintenant définir les opérations d'union et d'intersection de deux multi-ensembles. Proposez des implantations pour ces deux opérations.

```
multiset intersection(multiset b);
multiset reunion(multiset b);
```

Question 8 Déclarez, puis programmez une fonction amie de saisie au clavier des éléments d'un multi-ensemble. On pourra par exemple demander au programme d'énumérer un à un les codes ASCII et saisir pour chacun d'entre eux leur nombre d'occurences dans le multi-ensemble.

2 Représentation plus compacte

On souhaite représenter les multi-ensembles sans devoir stocker tous les zéros inutiles mais présents dans la représentation précédente. Pour cela, on va utiliser une liste de couples de la forme (code ASCII, nombre d'occurrences). Dans cette liste, seuls les codes ASCII dont le nombre d'occurences est non nul seront présents.
Question 9 Comment représenteriez-vous les multi-ensembles de la question 1 en utilisant cette nouvelle représentation ?
Question 10 On souhaite représenter les multi-ensembles par des listes chaînées de paires (code ASCII, nombre d'occurrences). Proposez des structures de données et une déclaration de classe `multiset_compact` permettant de faire cela. Comment faire pour construire le multi-ensemble vide ? Quelles conventions peut-on adopter pour que la recherche et l'insertion d'un nouveau caractère dans le multi-ensemble soit plus efficace ?
Question 11 Programmez les méthodes de base `add`, `remove`, `nb_occ` et `appartient` dans cette nouvelle représentation optimisée.