Programmation

– Examen (**1**) –

Jeudi, 19 Décembre 2024
Durée: 3 heures

Résumé

L'examen consiste à résoudre un petit problème en C. Vous disposez de fichiers de tests et d'un squelette de code pour démarrer. À la fin de l'examen, vous devez envoyer votre travail sous forme d'une archive `tar.gz` contenant le code source répondant au problème de l'examen à l'adresse email `<emmanuel.fleury@u-bordeaux.fr>`. L'archive devra avoir la forme suivante (remplacez les mots entre `<... >` par votre nom/prénom) :

```
<NAME>_<Forename>-exam/
    +- xmas-candies.c
    +- Makefile
```

Vous pouvez laisser vos fichiers de tests dans l'archive. Cela n'a aucune importance pour l'évaluation.

1 Cadeaux de Noël

Cette année, le Père Noël a décidé de changer de méthode pour répartir les friandises dans les paquets cadeaux. Il rassemble ses lutins dans le grand atelier du pôle Nord et leur expliquer ce qu'il a en tête :

> « Oh, oh, oh ! Mes chers lutins, cette année, vous mettrez deux friandises dans le plus de paquets possibles mais en faisant attention à ce que le nombre de chaque type de friandise en stock ne dépasse jamais la moitié du nombre total de friandises restantes à distribuer. »

Sur ces mots, le Père Noël retourne préparer sa longue nuit. Mais, les lutins ne voient pas du tout comment appliquer cette méthode. Après plusieurs jours de procrastination, ils décident de faire appel à vous pour les aider à concevoir un programme qui résoudra le problème. Vous êtes leur dernier espoir !

Mathématiquement, cela veut dire que si N est le nombre de types de friandises et v_i le nombre de friandises restantes de type i, nous voulons qu'à tout moment on vérifie :

$$\forall i \in \{1, \ldots, N\}, v_i \leq \frac{\sum_{j=1}^{N} v_j}{2}$$

Les cas à traiter sont donnés sous forme de fichiers texte qui suivront le format suivant :

— **Input** : La première ligne donne le nombre de cas à traiter, `T`. Chaque cas est donné sous la forme de deux lignes. La première ligne d'un cas contient le nombre de types de friandises, `N` et la seconde contient `N` entiers, `V1`, `V2`, ..., `VN`, où `Vi` représente le nombre de friandises de type i restantes.

— **Output** : Pour chaque cas, il faut afficher le numéro du cas ("`Case #x: `", où `x` est le numéro du cas traité) suivi de la répartition des friandises dans les paquets. La répartition est une séquence d'une ou deux lettres représentant les types de friandises à mettre dans le paquet. Mais, on ne peut avoir qu'un paquet au plus avec une seule friandise.

— **Limitations** : Aucun cas ne contiendra un type de friandise qui dépasse la moitié du stock total. Le nombre de types de friandises est au maximum de 26. Et, enfin : $1 \leq \texttt{T} \leq 50$, $1 \leq \sum_i \texttt{Vi} \leq 10000$.

Input

```
4
3
1 2 2
2
3 3
4
1 1 2 1
3
1 1 2
```

Output

```
Case #1: B CA BC
Case #2: AB AB AB
Case #3: C AB CD
Case #4: CA BC
```

Voici un exemple de fichiers d'Input et d'Output. Notez que souvent plusieurs réponses peuvent être possibles. Si vous voulez vérifier votre programme, un script Python est disponible dans l'archive de l'examen. Il est accompagné de plusieurs fichiers de tests (`xmas-candies-xsmall.in` (extra-small), `xmas-candies.in` (small), `xmas-candies-large.in` (large)).

Questions

1. Écrivez le parseur d'arguments qui permette de récupérer le nom du fichier d'entrée. En cas de problème, le programme doit afficher un message d'erreur sur `stderr` et retourner `EXIT_FAILURE`. Si le fichier est correctement ouvert, le programme doit renvoyer les solutions sur `stdout`. Voici un exemple d'usage du programme :

   ```
   $> ./xmas-candies
   xmas-candies: error: no input file given!
   $> ./xmas-candies xmas-candies-xsmall.in
   Case #1: B CA BC
   Case #2: AB AB AB
   Case #3: C AB CD
   Case #4: CA BC
   ```

2. Écrivez la partie du programme qui récupère les données des cas à traiter dans le fichier d'entrée. Nous supposerons que le fichier d'entrée est correctement formé, sans erreur de syntaxe, ni oubli de la part de l'utilisateur. Inutile, donc, de passer du temps à essayer d'être robuste lorsque vous parsez le contenu du fichier. Par contre, on s'attend à ce que vous détectiez l'absence du fichier sur les arguments, ou un problème lors de l'ouverture du fichier.

   Fonctions : `fscanf()`.

3. Écrivez le reste du programme en suivant les spécifications données au début du sujet. La clarté du code, les commentaires et l'efficacité du programme seront pris en compte. Assurez-vous aussi que l'usage de la mémoire soit correct. Tous ces aspects sont importants pour la notation finale.