Département d'Informatique
LICENCE 2ème année
Mardi 22 janvier 2008

Contrôle terminal de Programmation Fonctionnelle

Durée : 2h. Documents autorisés : une page de notes personnelles. On doit utiliser le langage OCAML et on ne doit pas utiliser d'éléments impurs.

I Récursivité et schémas de programmes

On suppose prédéfinis les schémas `List.map` et `List.fold_left` vérifiant :

```
List.map f [a1; a2; ...; an] = [f a1; f a2; ...; f an]
List.fold_left f a [b1; ...; bn] = f (... (f (f a b1) b2) ...) bn
```

On considère un message chiffré par un chiffrement de César : chaque lettre du texte en clair est remplacée par une autre lettre à distance fixe dans l'alphabet. Par exemple, si l'on utilise un décalage de 3, `'A'` serait remplacé par `'D'`, `'B'` deviendrait `'E'`, etc. Un tel message est donnée par une liste de mots. Chaque mot est une liste de lettres. Par exemple :

```
let texte = [['P';'I']; ['W';'S';'P';'I';'M';'P']; ['W';'I'];
             ['P';'I';'Z';'I']; ['E']; ['P']; ['I';'W';'X']];;
```

1. Définir la fonction `ajoute_occ` qui, étant données une liste de couples (lettre, nombre d'occurrences) et une lettre, ajoute un au nombre d'occurrences de la lettre dans la liste.

   ```
   ajoute_occ [('A',1);('B',3)] 'A' = [('A',2);('B',3)]
   ajoute_occ [('A',2);('B',3)] 'C' = [('A',2);('B',3);('C',1)]
   ```

2. Définir la fonction `repartion_occ` qui, étant donné un message, retourne une liste de couples (lettre, nombre d'occurrences).

   ```
   repartition_occ texte =
   [('P',5);('I',6);('W',3);('S',1);('M',1);('Z',1);('E',1);('X',1)]
   ```

   Utilisez la fonction `ajoute_occ` et le schéma `List.fold_left`.

3. Définir la fonction `repartion_freq` qui, étant donnée une liste de couples (lettre, nombre d'occurrences), retourne la liste de couples (lettre, fréquence d'apparition) où la fréquence d'apparition d'une lettre est son nombre d'occurrences divisé par le nombre total de lettres.

```
repartition_freq [('P',5);('I',6);('W',3);('S',1);
                  ('M',1);('Z',1);('E',1);('X',1)] =
[('P',0.263157894736842091); ('I',0.315789473684210509);
 ('W',0.157894736842105254); ('S',0.0526315789473684181);
 ('M',0.0526315789473684181);('Z',0.0526315789473684181);
 ('E',0.0526315789473684181);('X',0.0526315789473684181)]
```

Utilisez la fonction `float_of_int` de conversion d'un entier en réel.

4. Définir la fonction `trouve_cle` qui, étant donné un message chiffré avec le chiffrement de César, retourne le décalage du chiffrement de César. Ce décalage est la distance entre la lettre la plus fréquente dans le message et la lettre `'E'` (lettre la plus fréquente dans un texte en français).

```
trouve_cle texte = 4
```

Utilisez la fonction `Char.code` qui donne le code ASCII d'un caractère.

5. En déduire la fonction `dechiffre` qui, étant donnés un message chiffré et la clé, déchiffre le message.

```
dechiffre texte 4 = [['L';'E'];['S';'O';'L';'E';'I';'L'];['S';'E'];
                     ['L';'E';'V';'E'];['A'];['L'];['E';'S';'T']]
```

Utilisez le schéma `List.map` et la fonction `Char.chr` qui retourne un caractère étant donné son code ASCII.

II Lambda-calcul

Réduire ces λ-expressions en utilisant l'ordre normal de réduction (NOR). Indiquer le redex avec le symbole $_\lambda$. Préciser la règle utilisée (α- ou β-conversion) et utiliser la α-conversion lorsque cela est strictement nécessaire (en cas de problème de capture de variable).

1. $\lambda f.\lambda g.\lambda x.((f\ x)\ (g\ x))\ \lambda x.\lambda y.x\ \lambda x.\lambda y.x$
2. $\lambda f.\lambda g.\lambda x.((f\ x)\ (g\ x))\ \lambda x.\lambda a.(x\ a)\ \lambda a.\lambda x.(a\ x)$
3. $\lambda f.\lambda g.\lambda x.((f\ x)\ (g\ x))\ \lambda a.\lambda x.(a\ x)\ \lambda x.\lambda a.(x\ a)$