Examen médian IA01 A10
Université de Technologie de Compiègne

Durée : 2H
Les documents ne sont pas autorisés
Tous ordinateurs, toutes communications sont interdits

Utilisez trois copies séparées :
- *une copie pour la partie I*
- *une autre copie pour la partie II*
- *une autre copie pour la partie III*

1. Qu'avez-vous retenu du cours (5 points) ?

a) A quoi sert la représentation des connaissances ? Donnez deux formalismes de représentation.
b) Donner les composants d'un système expert et préciser leur rôle et/ou caractéristiques.
c) Donner les principales caractéristiques d'une programmation fonctionnelle. Donner un exemple de langage fonctionnel.
d) Qu'est-ce qu'un frame ? Quelles relations entretient-il avec un réseau sémantique ?
e) Qu'est-ce qu'une paire pointée ? Donnez sa structure interne, précisez comment la construire et la manipuler (car, cdr).

2. Recherche dans un espace d'états (8 points)

On considère le problème suivant dans lequel on doit passer de la situation de gauche à la situation de droite :

les seuls coups autorisés étant les retournements simultanés de deux flèches adjacentes.

On se propose de simuler la résolution de ce problème en utilisant une recherche dans un espace d'états.

Questions :

a) Qu'est-ce qu'un état ? Quel est l'état initial ? Quel est l'état final ? *(1 point)*
b) Proposer une représentation d'un état. On pourra coder chaque flèche par un chiffre en fonction de son sens. *(1 point)*

c) Quels sont les opérateurs permettant de passer d'un état à l'autre ? Quels sont les états successeurs de l'état initial ? Que se passe-t-il si on applique deux fois de suite le même opérateur ? *(1,5 point)*

d) Ecrire un algorithme permettant d'effectuer une recherche en profondeur d'abord. On pourra supposer que l'on dispose d'une fonction `successeurs` qui retourne la liste des états successeurs d'un état donné. *(2 points)*

e) Ecrire un algorithme pour la fonction `successeurs`. *(1 point)*

f) La recherche en profondeur n'est pas très efficace. On se propose de l'améliorer en choisissant, à chaque étape, parmi les états successeurs, l'état « le plus proche » de l'état final. Il faut pour cela disposer d'une fonction d'évaluation de la proximité d'un état par rapport à l'état final.

Définir une telle fonction d'évaluation et donner le code Lisp associé. *(1,5 point)*

3. Programmation Lisp (7 points)

On imagine créer un éditeur de texte. A partir d'une liste contenant les titres de chapitre et le texte, on souhaite réaliser un programme qui puisse mettre en forme le contenu de cette liste.
Nous faisons le choix de ne pouvoir mettre du texte que dans les feuilles.
Ainsi, nous souhaitons pouvoir représenter ce type de contenu :

```
Titre0
 Titre1
  Mon texte 1
 Titre2
  Titre2.1
   Titre2.1.1
    Mon texte 2.1.1
   Titre2.2.1
    Mon texte 2.2.2
  Titre2.2
   Mon texte 2.2
```

La représentation choisie pour ce type de contenu est une liste dont le premier élément contient le titre et les éléments suivants les sous-chapitres, et ce récursivement. Pour le contenu précédent, la représentation est :

```
(setq Text '("Titre0"
          ("Titre1"
           "Mon texte 1")
          ("Titre2"
           ("Titre2.1"
            ("Titre2.1.1"
             "Mon texte 2.1.1")
            ("Titre2.2.1"
             "Mon texte 2.2.2"))
           ("Titre2.2"
            "Mon texte 2.2"))))
```

a) Ecrire une fonction lisp Compte qui compte le nombre de chapitres. La fonction prend en entrée le texte et le niveau initial de titre (i.e. 0), elle renvoie le nombre de chapitres sous forme d'une variable non globale (nb : pour l'exemple précédemment présenté, le résultat serait 7) *(2 points)*
b) Ecrire une fonction lisp Profondeur qui calcule la profondeur maximale des chapitres. La fonction prend en entrée le texte, elle renvoie la profondeur maximale du texte (nb : pour l'exemple précédemment présenté, le résultat serait 4) *(2 points)*
c) Ecrire une fonction Affiche qui permet d'afficher le contenu d'un texte indenté correctement. Il s'agit de mettre une tabulation supplémentaire à chaque fois que l'on rentre dans un sous-chapitre (voir contenu présenté au début de l'exercice). La fonction prend en entrée le texte.
   Pour indenter le texte vous pouvez utiliser le ~t dans la fonction format et l'appeler autant de fois qu'il est nécessaire : *(format t "~t") (2 points)*
d) Ecrire une fonction lisp Rechercher qui permette de rechercher un texte donné. La fonction prend en entrée le texte complet, le texte recherché. Elle renvoie une valeur booléenne (true si le texte est trouvé, false s'il n'est pas trouvé) *(1 point)*