Université de Technologie de Compiègne
Durée : Deux Heures
Les documents ne sont pas autorisés
Tous ordinateurs, toutes communications sont interdits

Utilisez deux copies séparées :
- une copie pour les parties I et II
- une autre copie pour la partie III

I [6 points] - Qu'avez-vous retenu du cours ?

A. Donnez le principe des algorithmes génétiques, précisez-en le fonctionnement simplifié.
B. Citez les processus utilisés lors du raisonnement mis en œuvre dans le cadre d'un système RàPC. Précisez en quoi ces processus consistent.
C. En quoi consiste le pattern matching ? Donner la traduction française du terme. Précisez son analogie avec l'unification.
D. A quoi sert la représentation des connaissances ? Donnez deux formalismes de représentation.
E. Qu'est-ce que le perceptron ? Donnez sa composition.
F. Qu'appelle-t-on programmation fonctionnelle ?

II [7 points] - Polynômes à une variable

On choisit de représenter un polynôme à une variable par une liste ordonnée de coefficients dans l'ordre croissant des puissances de la variable, par exemple :

```
(setq p1 '(1 -2 3 1)) ; 1 - 2X + 3X^2 + X^3
(setq p2 '(1 0 -1))   ; 1 - X^2
(setq p3 '(-4 2 1))   ; -4 + 2 X + X^2
```

Si p et q sont deux polynômes soient les fonctions `ap` et `rlz` :

```
(defun ap (p q &optional r)
  (if (and p q)
    (ap (cdr p)(cdr q)(cons (+ (car p) (car q)) r))
    (if p
      (ap (cdr p) q (cons (car p) r))
      (if q
        (ap p (cdr q) (cons (car q) r))
        (reverse (rlz r))))))

(defun rlz (p)
  (if (zerop (car p))
    (if (cdr p) (rlz (cdr p)) p) p))
```

1) Détailler l'algorithme correspondant à la fonction ap **[2 points]**

2) Ecrire une fonction mbcp qui multiplie un polynôme par une constante **[2 points]**.

Exemple :

```
? (mbcp 0 p1)
(0)
? (mbcp 1 p1)
(1 -2 3 1)
? (mbcp -3 p1)
(-3 6 -9 -3)
```

3) En utilisant les fonctions ap et mbcp, écrire une fonction, mp, qui multiplie deux polynomes **[2 points].**

Exemple :

```
? (mp p1 p2)
(1 -2 2 3 -3 -1)
```

4) Représentation d'un polynôme creux **[1 point].**

Un polynôme creux est un polynôme du type : `X^120 + 5 X^25 + 2`
Quelle représentation pourrait-on choisir pour travailler avec ce genre de polynômes ?

III [7 points] Frames

On rappelle qu'un frame peut être représenté par une structure de données à trois niveaux dont le premier sert à repérer le *frame*, le suivant (*slot*) à décrire une propriété, et le troisième (*facettes*) à donner des informations plus précises sur la propriété en question. On peut spécifier la structure de données correspondante comme suit :

```
<frame> ::= (<frame-name> {<slot> }+)
<slot>  ::= (<slot-name> {<facet>}+)
<facet> ::= (<facet-name> <value>) | ( IF-ADDED <demon-name>)
                                   | ( IF-REMOVED <demon-name>)
                                   | ( IF-NEEDED <demon-name>)
```

où + indique une occurrence ou plus.

Dans un système de frames, les facettes sont définies une fois pour toutes, alors que les slots et les frames eux-mêmes sont définis par l'utilisateur en fonction des besoins de la représentation.

On pourra dans ce qui suit faire l'hypothèse que les facettes utilisées dans ce système ont pour nom : VALUE, DEFAULT, IF-ADDED, IF-REMOVED, IF-NEEDED, AUTHOR, DATE.
Par ailleurs, on distingue deux types de frames, le premier représentant des *concepts*, le second représentant des *individus* (manifestations de ces concepts).

Exemples :

```
(person
    (type ($value concept))
    (name ($default "Dupond "))
    (age ($if-added check-age))
    (sex  ($if-needed deduce-sex))
    (wife ($if-added inverse-status))
    (husband ($if-added inverse-status)) )
```

```
(person-12
    (type ($value individu))
    (name ($value "Martin"))
    (is-a ($value person))
    (age ($value 32))
    (wife ($value person-8)) ; la femme de person-12 est person-8
 )
```

On ne s'intéressera pas ici aux relations d'héritage entre concepts.

1) Création d'un individu [2 points]

Donner l'algorithme de la fonction *make-individual* permettant de créer un individu à partir d'un concept.
Exemple :

```
>  (make-individual 'person 'name "Martin" 'age 32)
person-12
```

2) Accès à l'information [2 points]

Ecrire le code Lisp de la fonction *getv* permettant d'obtenir la valeur d'une propriété pour un individu donné.
Exemples :

```
> (getv 'person-12 'name)
"Martin"
> (getv 'person-12 'sex)
M
; (M pour masculin, F pour féminin)
```

On pourra écrire pour cela une fonction de service *get-slot-facet* qui récupère une facette d'un slot pour un frame donné.
Exemples :

```
> (get-slot-facet 'person-12 'age '$value)
32
>(get-slot-facet ‘person ‘sex ‘$if-needed)
deduce-sex
```

3) Définition des démons [2 points]

a) Ecrire le code Lisp du démon *deduce-sex*, qui permet de déduire le sexe d'un individu si la valeur d'une des propriétés ‘wife’ ou ‘husband’ est connue.

b) Ecrire le code Lisp du démon *inverse-status* qui traduit le fait que si un individu x a pour femme y, alors y a pour mari x, et inversement.

4) Modification de la valeur d'une propriété [1 point]

Donner l'algorithme de *putv*, une fonction permettant de modifier la valeur d'une propriété
Exemple :

```
(putv 'person-12 'age 34)
```