Université Grenoble Alpes
MIAGE L3 – Programmations déclaratives

Contrôle continu
Février 2024

Durée : 80 minutes. 1 document personnel manuscrit A4 autorisé. Sujet sur 1 recto simple.

I. Exécutions et arbre d'exécution (barème indicatif : 6 points)

La spécification du programme suivant a été perdue.

```
equivalent([ ],[ ]).
equivalent([E|L],[E|R]) :- equivalent(L,R).
equivalent([2,E|L],[E|R]):-dif(E,2), equivalent([2|L],R).
equivalent([2,2|L],[4|R]):-equivalent(L,R).
```

Q1. Indiquez le/s résultat/s de 2 des 3 requêtes suivantes (ne pas justifier) :

```
?- equivalent([1,2,2,3,3], R).
?- equivalent(L,[4,2,1]).
?- equivalent(L,[X]).
```

Q2. Dessinez l'arbre d'exécution de la requête non abordée en Q1.

II. Regrouper les doublons (barème indicatif : 8 points)

L'objectif de cet exercice est de regrouper les doublons présents dans une liste de constantes.
Ex. : avec la liste en entrée d b h d h f g b h f après regroupement des doublons au niveau du premier doublon, la liste obtenue est : d d b b h h h f f g

Q3 Conception. Expliquer comment vous allez résoudre le problème par programme.
Q4 Spécification. Spécifier le/s prédicat/s de votre/s programme/s.
Q5 Réalisation. Donner le/s programme/s.

III. Inversion à commenter (barème indicatif : 6 points)

L'inverse d'une liste est la liste lue à l'envers, à partir de la fin.
Exemple : la liste en entrée 6 5 5 3 5 a pour liste inverse : 5 3 5 5 6
Spécification du programme : inverse(L,R) vrai ssi L en entrée a pour inverse R en sortie.
Q Propriétés algorithmiques. Analyser les programmes A, B, C suivants, et pour 2 des 3 programmes, au choix (le choix des 2 programmes à commenter est à vous), donner leurs propriétés (correction, complétion, terminaison). Pour chaque propriété, donner une ligne d'explication.

Prog A :

```
inverse([ ], [ ]).
inverse([E], [E]).
inverse([E|L], [F|R]) :- dernier(F, L), dernier(E, R), inverse(L,R).

dernier(E,[E]).
dernier(E,[F|L]) :-dernier(E,L).
```

Prog B :

```
inverse([ ], [ ]).
inverse(L, R) :- separePremierDernierEtEntre(PremL, EntreL, DerL, L),
                 inverse(EntreL, EntreR), separePremierDernierEtEntre(DerL, EntreR, PremL, R).

separePremierDernierEtEntre(Premier, [ ], Dernier, [Premier,Dernier]).
separePremierDernierEtEntre(Premier, [E|Entre], Dernier, [Premier,E|L]) :-
        separePremierDernierEtEntre(Premier, Entre, Dernier, [Premier|L]) .
```

Prog C:

```
inverse(L, R) :- inv(L, [], R).

inv([ ], [ ], [ ]).
inv([E|L], T, R) :-inv(L, [E|T], R).
```