

Durée : 70 minutes, 1 document A4 autorisé, Sujet sur 1 recto simple.

I. P ? (barème indicatif : 8 points)

Le programme suivant n'a pas été documenté, sa sémantique et son utilisation sont incertaines.

```
p(N, [], N) .
p(M, [pair|L], W) :-
    plusUn(N, M), foisDeux(W, V), p(N, L, V) .
p(M, [impair|L], W) :-
    foisDeux(N, M), plusUn(W, V), p(N, L, V) .

avec
plusUn(Q, P) :-
    {Q=P+1} .
foisDeux(W, V) :-
    {W=2*V} .
```

Q1 Exécution. Quelles seront la/les réponses aux requêtes suivantes (les explications ne sont pas nécessaires) :

```
?- p(3, [pair], R) .
?- p(5, [impair, impair], R) .
?- p(1, [A, B], N) .
?- p(N, [pair, pair, pair], 40) .
```

Q2. Arbre d'exécution et terminaison. Donner les (~10) premiers nœuds de l'arbre d'exécution de la requête suivante (les nœuds d'échec \perp comptent aussi) :

```
?- p(0, [impair|L], R) .
```

Cette requête termine-t-elle ? produit-elle des solutions ? si oui, lesquelles ?

II. Tri d'une liste de valeurs presque ordonnées (barème indicatif : 12 points)

Préambule : un ingénieur souhaite utiliser un tri adapté à ses données qui sont presque ordonnées (croissantes). L'idée est la suivante : dans un premier temps il découpe la liste initiale lors d'un parcours élémentaire pour diviser la liste initiale en sous-listes contiguës ordonnées les plus longues possible. Puis il fusionne par interclassement les listes ordonnées obtenues jusqu'à obtenir le résultat. Ex. pour trier la suite L : 1 8 44 67 4 40 73 7 0 9 55 1 6 16, il commence par diviser cette liste L en 5 sous-listes contiguës ordonnées L1, L2, ..., L5 (ici, cela donne 5 listes, dans le cas général, il faut prévoir un nombre quelconque de listes : il y aura une liste de listes) :

- L1 : 1 8 44 67
- L2 : 4 40 73
- L3 : 7
- L4 : 0 9 55
- L5 : 1 6 16 (en interne, le résultat ressemble plutôt à LL : [L1, L2, ..., L5], une liste de 5 listes triées)

Puis il fusionne par interclassement les 2 premières listes L1 et L2, cela donne 1 4 8 40 44 67 73. Il recommence avec cette nouvelle liste et L3 pour obtenir une nouvelle liste triée, et ainsi de suite jusqu'à fusionner par interclassement L5 et obtenir la liste complète triée.

Q1. Fusion simple par interclassement. Spécifiez et réalisez un prédicat qui fusionne par interclassement 2 listes d'entiers déjà ordonnées (croissantes) pour obtenir une liste triée par valeurs croissantes. Pour comparer 2 nombres, utiliser `plusGrand(A, B) :- {A>B}`. vrai ssi A est plus grand que B.

Q2. Fusions multiples par interclassement. Spécifiez et réalisez un prédicat qui utilise le prédicat de la question 1 pour fusionner successivement toutes les listes d'une liste de listes ordonnées et obtenir une liste triée par valeurs croissantes contenant tous les éléments de toutes les listes initiales. C'est la seconde étape de l'algorithme décrit en préambule.

Q3. Séparation en sous-listes ordonnées. Spécifiez et réalisez un prédicat qui sépare une liste en une liste de sous-listes ordonnées. C'est la première étape de l'algorithme décrit en préambule.

Q4. Tri de listes presque ordonnées. Spécifiez et réalisez un prédicat qui fait le lien entre les prédicats des