Examen de janvier 2007

durée 2h - documents non autorisés

Exercice 1 : *Les cartes*

Dans cet exercice, on considère une situation initiale des tas de cartes dans laquelle le tas n°1 contient un nombre quelconque de `T` et de `K`, mais en tout cas au moins un `T` qui peut être situé n'importe où.

Q 1 .

Parmi les initialisations du tas n°1 qui suivent une seule correspond à celle souhaitée. Laquelle ? Vous donnerez des réalisations effectives qui montrent que les autres initialisations sont incorrectes.

1. `initTas(1,'T[T]+[K]');`
2. `initTas(1,'[T+K]T[T+K]');`
3. `initTas(1,'(T+K)[T+K]');`

Q 2 . On souhaite que le robot place le dernier `T` (c'est–à–dire celui situé le plus bas) dans le tas n°1 sur le tas n°2, toutes les autres cartes se trouvant sur le tas n°3.

Q 2.1. Commencez par faire en sorte que le premier `T` se retrouve sur le tas n°2, les `K` initialement situés au dessus se retrouvant sur le tas n°3.

Q 2.2. Poursuivez le programme afin d'atteindre la situation souhaitée.

Q 3 . Si on suppose que le tas n°3 est initialement vide comment programmer le robot pour qu'il remette toutes les cartes du tas n°3 sur le tas n°1 ?

Q 4 . On suppose maintenant que le tas n°3 n'est pas nécessairement vide.

Q 4.1. Que faut-il ajouter au robot afin qu'il puisse remettre sur le tas n°1 toutes les cartes du tas n°3 initialement sur le tas n°1 ?

Q 4.2. Reprogrammez le robot en conséquence, et déclarez les variables que vous utilisez.

Q 5 . Réalisez une procédure nommée `deplacerDernierTrefle` à trois paramètres `i,j,k` qui déplace le dernier `T` du tas `i` vers le tas `j` en se servant du tas `k` comme tas intermédiaire, les trois tas étant supposés distincts. Précisez les autres contraintes d'utilisation de cette procédure.

Q 6 . Dans cette dernière question, on suppose que tous les tas sont initialisés comme il a été décrit au départ pour le tas n°1. Utilisez la procédure réalisée précédemment pour arriver à une situation finale où toutes les cartes sont situées dans leur tas initial, sauf les quatre `T` les plus bas de chaque tas qui se trouvent au sommet du tas n°1.

Exercice 2 : *Une transformation*

Voici une fonction écrite en PASCAL qui transforme un nombre entier en un autre.

```
function transforme(n : CARDINAL ) : CARDINAL;
var
   r,m : CARDINAL;
begin
   r := 0;
   m := n;
   while m<>0 do begin
      r := r*10 + (m mod 10);
      m := m div 10;
   end {while};
   transforme := r;
end {transforme};
```

On rappelle que les opérateurs **mod** et **div** calculent respectivement le reste et le quotient de la division euclidienne des entiers.

Q 1 .

Calculez `transforme(5103)` en présentant sous forme d'un tableau la valeur des variables `r` et `m` à la fin de chaque étape de la boucle.

Q 2 . Écrivez le corps d'un programme qui demande à l'utilisateur deux entiers positifs a et b, puis affiche à l'écran les valeurs de la fonction `transforme` pour tous les entiers de a à b, à raison d'une valeur par ligne. L'affichage d'une ligne se fera sous la forme

`5103 : 3015`

Vous n'oublierez pas de déclarer les variables nécessaires.

Exercice 3 : *Sur les chaînes*

Dans cet exercice on réalise une fonction `crochete` paramétrée par une chaîne de caractères `s` et dont le résultat est une chaîne de caractères, obtenue en entourant chacun des caractères de la chaîne `s` par des crochets. Par exemple `crochete('timoleon')` vaut `'[t][i][m][o][l][e][o][n]'`

Q 1 . En supposant déjà écrite la fonction `crochete`, expliquez comment utiliser cette fonction pour initialiser le tas 1 avec un nombre quelconque de trèfles surmontés d'un nombre quelconque de carreaux surmontés d'un nombre quelconque de coeurs, surmontés d'un nombre quelconque de piques ?

Q 2 . Réaliser une fonction nommée `encadre` paramétrée par un caractère `c` dont le résultat est une chaîne de trois caractères, composée d'un crochet ouvrant, du caractère `c` et d'un crochet fermant.

Q 3 . En utilisant la fonction `encadre` écrire la fonction `crochete`

Q 4 . Au vu des instructions qui suivent, déclarez les variables `s` `c`, `res` et `i`.

```
s:='[t][i][m][o]';
res:='';
for i:=1 to length(s) do
begin
  if (i mod 3)=2 then
  begin
    c:=s[i];
    res:=res+c;
  end {if};
end {for};
```

Q 5 . Précisez le contenu de chaque variable après l'exécution des instructions précédentes.

Q 6 . Implantez une fonction `decrochete` telle que pour toute chaîne `s` on ait `decrochete(crochete(s))=s`