Examen de mars 2007 (2nde session)

durée 2h - documents, calculatrices, portables non autorisés

Exercice 1 : *Les cartes*

Q 1 . Parmi les initialisations du tas 1 suivantes, lesquelles garantissent que le tas 1 contient au moins deux cartes ? Pour les autres, donnez des contre exemples.

1. `InitTas(1,'T[K+P]T');`
2. `InitTas(1,'T[T]+TP');`
3. `InitTas(1,'T(T[T]+TP)');`
4. `InitTas(1,'[TP]');`
5. `InitTas(1,'T(K+P)T');`
6. `InitTas(1,'T+P');`

Dans la suite du sujet, on suppose que le tas 1 a été initialisé avec la première instruction.
On suppose que les autres tas sont vides.

Q 2 . Donner une séquence d'instructions qui permet de passer de la situation initiale à la situation suivante :

//tas 1:'T[K+P]' tas 2:'T' tas3 :'' tas 4:''

Q 3 . Dans cette question **uniquement**, on suppose qu'initialement, il n'y a qu'une seule carte entre les deux trèfles du tas 1. Donner une séquence d'instructions qui permet de passer de la situation initiale à la situation suivante :

//tas 1:'' tas 2:'TT' tas3 :'[K]' tas 4:'[P]'

Q 4 . Donner une séquence d'instructions qui permet de passer de la situation initiale à la situation suivante :

//tas 1:'' tas 2:'TT' tas3 :'[K]' tas 4:'[P]'

Q 5 . Donner une séquence d'instructions qui permet de passer de la situation initiale à la situation suivante :

//tas 1:'' tas 2:'TT' tas 3 :'[K+P]' tas 4:''

avec la condition supplémentaire que le tas 2 soit trié dans l'ordre croissant des cartes.

Q 6 . Comment passer de la situation précédente, à la situation suivante :

//tas 1:'[K+P]T[K+P]T[K+P]' tas 2:'' tas 3:'' tas 4:''

avec les conditions supplémentaires suivantes :

- les cartes situées sous le trèfle le plus bas sont strictement inférieures à ce trèfle ;
- les cartes situées entre les deux trèfles ont une hauteur comprise au sens large entre ces deux trèfles ;
- et enfin, les cartes situées au dessus du second trèfle ont une hauteur supérieure stricte à la hauteur de ce trèfle.

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*100 + (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 : 3000105
```

Vous n'oublierez pas de déclarer les variables nécessaires.

Exercice 3 : *Sur les chaînes*

Dans cet exercice, on considère les adresses électroniques de la forme

`prenom.nom@fournisseur.domaine`

Par exemple, dans l'adresse `Raymond.Calbuth@etudiant.univ-lille1.fr`, le prénom est `Raymond`, le nom est `Calbuth`, le fournisseur est `etudiant.univ-lille1` et le domaine est `fr`.

Comme l'exemple le montre, le fournisseur peut contenir un ou plusieurs points. On fera l'hypothèse que les prénoms, noms et domaines ne peuvent pas en contenir.

Les adresses électroniques sont représentées par des chaînes de caractères.

Q 1 . Réalisez une fonction qui construit une adresse électronique à partir de ses quatre éléments constitutifs.

Q 2 . Réalisez une fonction qui extrait le prénom à partir d'une adresse électronique.

Q 3 . Réalisez une fonction qui extrait le nom à partir d'une adresse électronique.

Q 4 . Réalisez une fonction qui extrait le fournisseur à partir d'une adresse électronique. Attention, cette question bien que ressemblant à la précédente ne se résoud pas de la même manière.