Examen structures de données
Janvier 2004

CORRIGE SUCCINCT

Problème I
Manipulation de liste

Question 1.

```
max(L : liste) : élément ;
élément x ;
début
        si est_listevide(L)
                alors retourner (faux)
                sinon   p :=premier(L) ;
                        x :=contenu(p,L) ;
                        tant que non est_dernier(p)
                        faire
                                si contenu(p,L) > x
                                        alors x := contenu(p,L)
                                finsi
                                p :=suivant(p)
                        fintantque
        finsi
        retourner(x)
fin
```

Question 2.

max est un observateur car il retourne un élément extérieur au type liste.

Question 3.

```
Type triplet ;
Type place is access triplet ;
Type triplet is
        Record
                Contenu : element;
                Precedent : place;
                Suivant : place;
        End record;
Type liste is
        Record
                Tete : place := NIL;
        End record ;
Procedure supprimer_max(L : liste) is
        P : place := L.Tete;
```

```
        Trouve : Boolean ;
Début
        Trouve := faux ;
        Si place < > NIL
                Alors
                        Tant que non Trouve
                                Faire
                                        Si contenu = max(L)
                                        Alors
                                                Trouve := vrai ;
                                                P.Precedent.Suivant := P.Suivant;
                                                P.Suivant.Precedent := P.Precedent ;
                                        Finsi
                                P := P.Suivant;
                        Fintantque
        Finsi
Fin
```

Problème II
Fonctionnement d'un ascenseur

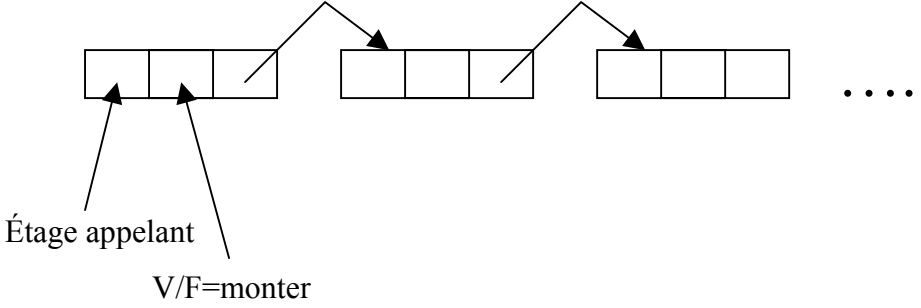
Stratégie :

- **A priori, répondre dans l'ordre chronologique des appels**
- **Si, au passage, on peut optimiser en s'arrêtant à un étage parce qu'un appel a été émis depuis cet étage pour aller dans la même direction, s'y arrêter.**

Je propose la combinaison de deux structures :

- **une file (liste chaînée) avec, à chaque position :**
  - **l'étage appelant (nb entier),**
  - **le sens de l'appel (booléen),**
  - **un pointeur vers le suivant**
- **une structure de tableau avec, pour chaque étage, une indication, selon qu'il y a eu un appel ou non en provenance de cet étage et que cet appel n'a pas été traité.**

Grâce à la première structure, on peut traiter les appels dans leur ordre d'arrivée. Avec la seconde, on peut d'une part s'arrêter au passage pour optimiser et on peut d'autre part, par exemple, quand on est au $7^{ème}$, monter au $8^{ème}$ si un appel en émane avant de redescendre :



	aucun	monter	monter	aucun	descend	Les 2	descend	aucun
Etage	1	2	3	4	5	6	7	8