Contrôle continu 2009-2010
INF251
Pointeurs - Récursivité – Listes – Piles - Files

Questions de cours (6 points)

- L'algorithme de la primitive défiler doit-elle contenir l'appel à fileVide ? Pourquoi ?
  Non. Dans des algorithmes, on pourrait être amené à tester si la file est vide avant de dépiler. Il y aurait donc un second test de fileVide au sein de la primitive. Les langages de programmations actuels permettent d'insérer un tel test qui pourra être supprimé dans une compilation finale (instruction assert en C par exemple)
- Donnez une définition du type pile en implémentation dynamique. Dans ce contexte, écrire la primitive dépiler.

```
Pile d'objet=listeSC d'objet ;
fonction dépiler(ref P:pile de objet):vide;
  début
    supprimerEnTete(P);
  fin
finfonction
```

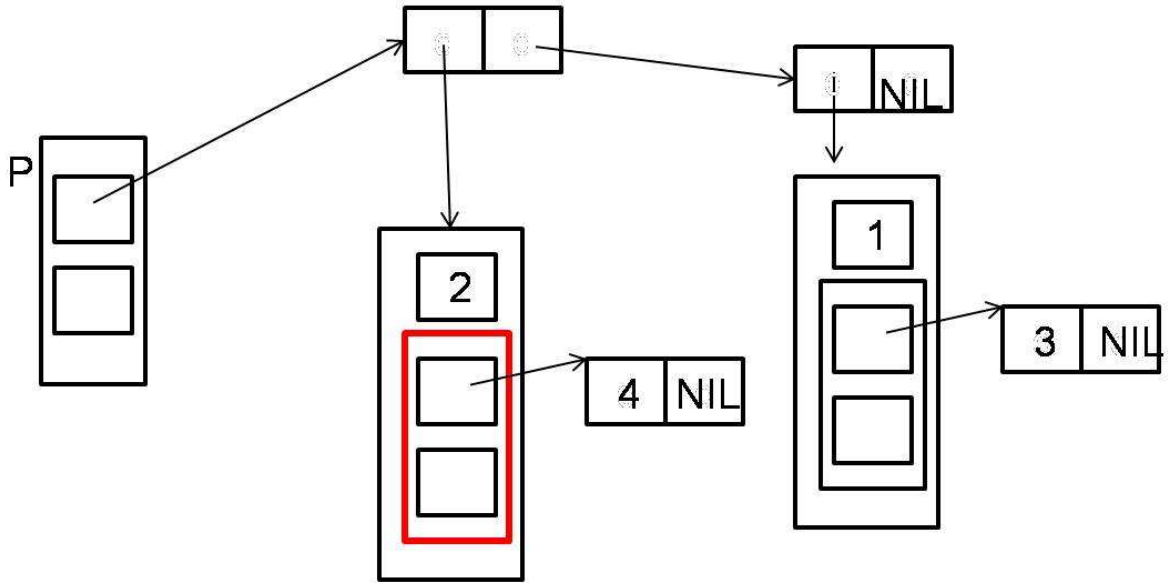
- Soit la fonction :

```
Fonction essai() : listeSC d'entier ;
  var t : ^tag ;
   type tag=structure
            v : entier ;
            s :listeSC d'entier ;
          finstructure
   var P :pile de ^tag ;
   var i :entier ;
   creerPile(P) ;
   pour i=1 à 2 faire
     new(t) ;
     creerListe(t^.s);
     t^.v=i;
     ajouterEnTete(t^.s, i+2)
     empiler(P,,t);
   finpour
   retourner(valeur(F)^.s) ;
finfonction
```

On considère que les piles et listes utilisent l'allocation dynamique. Dessinez la mémoire avant l'exécution de l'instruction retourner. Indiquez sur ce dessin la partie retournée par la fonction ? Pourquoi cette fonction est-elle mal écrite ?

On note d'abord que cette fonction ne retourne rien d'intelligible : il y a écrit valeur(F) or F n'existe pas ! Supposons donc que il y ait écrit `retourner(valeur(P)^.s) ;`

La partie rouge du dessin ci-dessous représente ce qui est retourné par la fonction. Cette fonction est mal écrite car elle produit une fuite mémoire : certaines cellules allouées dynamiquement resteront inaccessibles après exécution.



Exercice 1 (4 points)

Pour une file ayant N objets enfilés, peut-on connaitre le $p^{ième}$ élément après le premier? Si non, pourquoi ? Si oui, écrire l'algorithme regarderPieme qui réalise cette opération.

Le type abstrait file n'est pas conçu pour cela cependant en visitant une file temporaire copie de la file de départ, on peut extraire le $p^{ième}$ élément. Si l'implémentation est statique la fonction ci-dessous réalise l'opération demandée.

```
Fonction regarderPieme(val F :file d'objet ;val p :entier) :objet ;
  Var i : entier ;
  Début
  Pour i allant de 1 à p-1 faire
    Défiler(F) ;
   Finpour
  Retourner(valeur(F)
 Fin
```

Exercice 2 (10 points)

1 - Ecrire une fonction qui prend en entrée une pile d'entier et qui modifie la pile en supprimant les entiers pairs et fournit en sortie la liste des entiers pairs supprimés. Par exemple, la pile [1,4,5,2,3,6,8[(où 1 est le fond de pile) est transformée en la pile [1,5,3[où (1 est le fond de pile) et la liste fournie est (4,2,6,8), les entiers sont dans l'ordre de leur entrée dans la pile. Donnez la complexité de votre fonction.

```
Fonction triPair(ref P :pile d'entier) : listeSC d'entier ;
   var L :listeSC d'entier ;
   var P1 :pile d'entier ;
   var a :entier ;
   début
      créerPile(P1) ;
      creerListe(L) ;
      tantque !pileVide(P) faire
         a=valeur(P) ;
         depiler(P) ;
         si a est pair alors
```

```
            insererEntete(L,a) ;
        sinon
            empiler(P1,a) ;
        finsi
    fintantque
    tantque !pileVide(P1) faire
        empiler(P,valeur(P1)) ;
        depiler(P1)
    fintantque
    detruirePile(P1) ;
    retourner(L) ;
fin
```

2 - Que doit-on changer dans votre algorithme si on souhaite que l'ensemble des éléments de la liste de sortie soit (8,6,2,4), les entiers sont dans l'ordre inverse de leur entrée dans la pile. ?

```
Fonction triPair(ref P :pile d'entier) : listeSC d'entier ;
    var L :listeSC d'entier ;
    var P1 :pile d'entier ;
    var a=entier ;
    début
        créerPile(P1) ;
        creerListe(L) ;
        tantque !pileVide(P) faire
            a=valeur(P) ;
            depiler(P) ;
            si a est pair alors
                si listeVide(L) alors
                    insererEntete(L,a) ;
                    debutListe(L) ;
                sinon
                    insererApres(L,a) ;
                    suivant(L) ;
                finsi
/* déjà présent dans le corrigé du DS1 */
            sinon
                empiler(P1,a) ;
            finsi
        fintantque
        tantque !pileVide(P1) faire
            empiler(P,valeur(P1)) ;
            depiler(P1)
        fintantque
        detruirePile(P1) ;
        retourner(L) ;
    fin
```

Liste Simplement Chainées

fonction valeur(val L:liste d'objet):objet;
fonction debutListe(ref L:liste d'objet):vide;
fonction suivant(ref L:liste d'objet):vide;
fonction listeVide(val L:liste d'objet): booleen;
fonction créerListe(ref L:liste d'objet):vide;
fonction insérerAprès(ref L:liste d'objet; val x:objet;):vide;
fonction insérerEnTete(ref L:liste d'objet ;val x:objet):vide;
fonction supprimerAprès(ref L:liste d'objet):vide;
fonction supprimerEnTete(ref L:liste d'objet):vide;
fonction detruireListe(ref L:lisetd'objet):vide;</pre>

Liste Doublement Chainées (ajout de)

fonction finListe(ref L:liste d'objet):vide;
fonction précédent(ref L::liste d'objet): vide;

Fonctions sur les listes

fonction estFinListe(val L:liste d'objet):booléen;
fonction appartient(ref L:liste d'objet; ref x:objet): booléen ;

Piles

fonction valeur(ref P:pile de objet):objet;
fonction fileVide(ref P:pile de objet):booléen;
fonction créerPile(P:pile de objet);
fonction empiler(ref P:pile de objet;val x:objet):vide;
fonction dépiler(ref P:pile de objet):vide;
fonction detruirePile(ref P:pile de objet):vide;

Files

fonction valeur(ref F:file de objet):objet;
fonction fileVide(ref F:file de objet):booléen;
fonction créerFile(F:file de objet);vide;
fonction enfiler(ref F:file de objet;val x:objet):vide;
fonction défiler (ref F:file de objet):vide;
fonction detruireFile(ref F:file de objet):vide;