Université Savoie Mont Blanc, UFR SceM
L3 S&T TRI/ESET

6 janvier 2017
Durée: 1h30

INFO510 – EXAMEN

Documentation autorisées : une feuille au format A4.

Directives :

- Les exercices sont indépendants et l'ordre n'est pas important.
- Tout le code demandé doit être rédigé dans le langage pseudo-code vu en cours.
- Vous pouvez utiliser sans les redéfinir tous les types de données abstraits vu en cours et en TD. Les fonctions pour les manipuler sont données aux dernières pages de cet énoncé.
- Vous pouvez toujours supposer les questions précédentes résolues, et donc utiliser les actions/fonctions précédentes comme si vous les aviez écrites correctement.

Question 1. Trace d'un programme (/5)

Déterminez ce qui est affiché par chacun des deux programmes ci-dessous.

a)

```
Variables : x, y : Chaine
            i, n, r : Entier
            a, b : Caractère
Début
    r := 0
    x := "Bonjour"
    y := "Pondeur"
    n := taille( x )
    Affiche ( "n=", n )
    Pour i de 0 à n − 1 faire
        a := getChar( x, i )
        b := getChar( y, i )
        Si a == b alors
            r := r + 1
            Affiche ( "a=", a, ", i=", i, ", r=",r )
        Fin si
    Fin pour
    Affiche ( "r=", r )
Fin
```

b)

```
Fonction s( x : E Entier, y : E Entier, z : ES Entier )
Début
    Affiche ( "x=", x, ", y=", y, ", z=", z )
    Si x > 0 alors
        z := z + 1
        s( x − 1, y, z )
    Sinon Si y > 0 alors
        z := z + 1
        s( x, y − 1, z )
    Fin si
Fin

Variable : t : Entier initialisé à 0
Début
    s( 2,1,t )
    Affiche ( "t=", t )
Fin
```

Question 2. Types de données complexes (/8)

Considérez les définitions suivantes :

Type : Évaluation : Entité
titre : Chaine
note : Flottant

Type : Étudiant : Entité
nom : Chaine
prénom : Chaine
évals : Tableau[0 .. 3] de Évaluation

Constante : NbEtu := 42

Type : Classe : Tableau [0..NbEtu-1] de Étudiant

Écrivez le pseudo-code des fonctions ci-dessous.

a) **Fonction** note(e : **E** Étudiant) : Flottant
Calcule la note finale d'un étudiant en faisant la moyenne des notes obtenues à chacune des évaluations.

Hypothèse : le paramètre e a été initialisé et contient des données valides.

b) **Fonction** top(C : **E** Classe)
Affiche le nom et le prénom de tous les étudiants de C ayant obtenu la meilleur note.

Contrainte : la note finale de chaque étudiant ne doit être calculée qu'une seule fois.

Hypothèse : le tableau C a été initialisé et contient des données valides.

Conseil : utilisez une des structure de données vues en cours pour stocker les indices de C où se trouvent les étudiants ayant obtenu la meilleure note. N'oubliez pas de redéfinir le type `Élément`.

Question 3. Implémentation d'un type abstrait (/8)

On définit le type abstrait **MinExtractor** de la manière suivante :

- Il s'agit d'un conteneur dont les valeurs contenues sont de type `Élément`.
- On suppose que le type `Élément` supporte les opérations de comparaison ($<$, $>$, $\leq$, $\geq$) et d'affectation (:=).
- La fonction `me_init` initialise un **MinExtractor** vide.
- La fonction `me_insère` insère une valeur dans un **MinExtractor**.
- La fonction `me_estVide` teste si un **MinExtractor** est vide ou non.
- La fonction `me_getMin` retourne la valeur minimum contenue dans un **MinExtractor** non-vide.
- La fonction `me_retireMin` retire la valeur minimum contenue dans un **MinExtractor** non-vide.

Voici un exemple d'utilisation :

```
Type: Élément : Entier
Variables : M : MinExtractor
            x : Entier
Début
   me_init( M )
   me_insère( M, 3 )
   me_insère( M, 5 )
   me_insère( M, 2 )
   me_insère( M, 3 )
   me_insère( M, 1 )
   Tant que non me_estVide( M ) faire
      x := me_getMin( M )
      me_retireMin( M )
      Affiche ( x )
   Fin tant que
Fin
```

Affichage :
1
2
3
3
5

a) Définissez le type **MinExtractor**. Les valeurs contenues dans un **MinExtractor** doivent obligatoirement être stockées dans un tableau de taille **MAX**. On suppose ici que **MAX** est une constante prédéfinie.

b) Implémentez les fonctions suivantes. Dans chaque cas, remplacez le symbole ? par le symbole le plus approprié parmi : E, S, ES.

Fonction me_init(M : ? MinExtractor)

Fonction me_insère(M : ? MinExtractor, x : ? Élément)

Fonction me_estVide(M : ? MinExtractor) : Booléen

Fonction me_getMin(M : ? MinExtractor) : Élément

Fonction me_retireMin(M : ? MinExtractor)

Annexe : Entêtes de fonctions

Les fonctions suivantes ont été vues en cours ou en TD. Vous pouvez les utiliser sans les redéfinir.
Attention : ne faites aucune hypothèse sur la manière donc sont implémentés ces types abstraits.

Chaines de caractères et caractères

Fonction getChar(c : **E** Chaine, *pos* : **E** Entier) : Caractère

Fonction setChar(c : **ES** Chaine, *pos* : **E** Entier, *valeur* : **E** Caractère)

Fonction taille(c : **E** Chaine) : Entier

Fonction init(c : **S** Chaine, n : **E** Entier, a : **E** Caractère)

Fonction sousChaine(c : **E** Chaine, *pos* : **E** Entier, *lng* : **E** Entier) : Chaine

Fonction concat($c1$: **E** Chaine, $c2$: **E** Chaine) : Chaine

Fonction asciiValue(x : **E** Caractère) : Entier

Fonction charValue(e : **E** Entier) : Caractère

Fonction carVersEntier(x : **E** Caractère) : Entier

Fonction LireEntier(c : **E** Chaine, *pos* : **E** Entier, *lng* : **E** Entier) : Entier

Files et piles

Fonction file_init(f : **S** File)

Fonction file_estVide(f : **E** File) : Booléen

Fonction file_enfile(f : **ES** File, e : **E** Élément)

Fonction file_premier(f : **E** File) : Élément

Fonction file_défile(f : **ES** File)

Fonction pile_init(p : **S** Pile)

Fonction pile_estVide(p : **E** Pile) : Booléen

Fonction pile_empile(p : **ES** Pile, e : **E** Élément)

Fonction pile_dessus(p : **E** Pile) : Élément

Fonction pile_dépile(p : **ES** Pile)

Ensemble

Fonction ens_init(e : **S** Ensemble)

Fonction ens_estVide(e : **E** Ensemble) : Booléen

Fonction ens_contient(e : **E** Ensemble, x : **E** Élément) : Booléen

Fonction ens_insère(e : **ES** Ensemble, x : **E** Élément)

Fonction ens_retire(e : **ES** Ensemble, x : **E** Élément)

Fonction ens_union($e1, e2$: **E** Ensemble) : Ensemble

Fonction ens_inter($e1, e2$: **E** Ensemble) : Ensemble

Fonction ens_diff($e1, e2$: **E** Ensemble) : Ensemble

Tableau associatif

Fonction ta_init(t : **S** TabAsso)

Fonction ta_insère(t : **ES** TabAsso, k : **E** Clé, v : **E** Valeur)

Fonction ta_retire(t : **ES** TabAsso, k : **E** Clé)

Fonction ta_contient(t : **E** TabAsso, k : **E** Clé) : Booléen

Fonction ta_valeur(t : **E** TabAsso, k : **E** Clé) : Valeur

Itérateur

On considère ici un itérateur défini sur un type `Conteneur` quelconque.

Fonction cnt_it_init(e : **E** Conteneur, it : **S** ConteneurIterateur)

Fonction cnt_it_suivant(e : **E** Conteneur, it : **ES** ConteneurIterateur) : Element

Fonction cnt_it_termine(e : **E** Conteneur, it : **E** ConteneurIterateur) : Booléen