DS2 - Algorithmes et Structures de Données

Mardi 11 Janvier 2022

1. Tableaux – 3 pts

Écrire en pseudo-langage une procédure `diff` qui prend en entrée deux tableaux d'entiers `t1` et `t2` (non triés et sans doublon) de taille respective `n1` et `n2` et qui renvoie le tableau `t3` de taille `n3` représentant la **différence symétrique** de `t1` et `t2` (ensemble des éléments appartenant soit à `t1`, soit à `t2`, **mais pas au 2**). Cette procédure peut appeler d'autres fonctions ou procédures que vous écrirez.

```
Const max = 30
Type tab = tableau[1..max] d'entier

Fonction appartient(e : entier, t : tab, n : entier) : booléen
Var i : entier
    trouve : booléen
Début
i← 1
trouve ← false
TantQue i<=n et not trouve Faire
      Si t[i]=e
            Alors trouve ← vrai
      FinSi
      i ← i+1
FinTantQue
retourner(trouve)
Fin

Procédure diff(E t1,t2 : tab, E n1, n2 : entier ; S t3 : tab, S n3 : entier)
Var i,j : entier
Début
n3← 1
{pour vérifier les éléments de t1}
Pour i← 1 à n1 inc +1 Faire
      Si not appartient(t1[i],t2,n2)
            Alors t3[n3]← t1[i]
                  n3 ← n3+1
      FinSi
FinPour
{pour vérifier les éléments de t2}
Pour i← 1 à n2 inc +1 Faire
      Si not appartient(t2[i],t1,n1)
            Alors t3[n3]← t2[i]
                  n3 ← n3+1
      FinSi
FinPour
n3 ← n3-1
Fin
```

2. Récursivité – 4 pts

Ecrire en pseudo-langage une **fonction récursive** qui renvoie le plus petit entier d'un tableau `t` non trié entre les indices `i` et `j` par la méthode de dichotomie : on sépare le tableau en 2 sous-tableaux à chaque appel de la fonction, et le plus petit entier du tableau est le min des plus petits entiers des 2 sous-tableaux.

```
Const max = 30
Type tab = tableau[1..max] d'entier
```

```
Fonction min(x,y : entier) : entier
Var  res : entier
Début
Si x<=y
      Alors res ← x
      Sinon res ←  y
FinSi
retourner(res)
Fin

Fonction ppentier(t : tab ; i,j : entier) : entier
Var  res, m : entier
Début
Si i=j
      Alors res ← t[i]
      Sinon m ← (i+j) div 2
            res ← min(ppentier(t,i,m),ppentier(t,m+1,j))
FinSi
retourner(res)
Fin
```

3. File de priorités – 4 pts

On souhaite gérer la file d'attente des documents à traiter par un service. Cette file est représentée par une liste chaînée de documents dont on connait le nom du fichier `nomdoc`, sa priorité `prio` (de 1 à 3 selon l'urgence du traitement du document, 1 étant le plus urgent), le type de traitement à effectuer `type` et le nom de l'utilisateur `util` qui demande ce traitement. Cette liste chaînée est toujours triée dans l'ordre croissant des priorités et, pour une même priorité, dans l'ordre des arrivées. Le premier document de la file (en tête) est le premier document arrivé avec la priorité 1 et le dernier (en queue) est le dernier arrivé avec une priorité 3.

```
Type  doc = Enregistrement
                  nomdoc, util, type : chaîne
                  prio : entier
                  suiv : ^doc
            FinEnregistrement
      file = Enregistrement
                  tete, queue : ^doc
             FinEnregistrement
```

Ecrire en pseudo-langage une procédure `ajoute` qui ajoute un document de nom `n`, de l'utilisateur `u`, de type `t` et de priorité `p`, dans la file `f`. **Attention aux cas particuliers et à l'optimisation de l'algorithme.**

```
Procédure ajoute(E n, u, t : chaine, E p : entier ; E/S f : file)
Var q,r : ^doc
Début
r←allouer(doc)
r^.nomdoc←n
r^.util←u
r^.type←t
r^.prio←p
Si f.tete=nil           {si la file est vide}
      Alors f.tete ← r
            r^.suiv ← nil
            f.queue ← r
      Sinon
            Si p≥f.queue^.prio {si la priorité est plus grande ou égale que
                                celle de la queue alors insertion en queue}
                  Alors f.queue^.suiv ← r
                        r^.suiv ← nil
                        f.queue ← r
                  Sinon Si p<f.tete^.prio  {si la priorité est strictement plus
                            petite que celle de la tete alors insertion en tete}
```

```
                                Alors r^.suiv ← f.tete
                                     f.tete ← r
                                Sinon q ← f.tete       {on cherche la bonne place}
                                     TantQue p≥q^.suiv^.prio et q^.suiv≠nil Faire
                                           q ← q^.suiv
                                     FinTantQue
                                     r^.suiv ← q^.suiv
                                     q^.suiv ← r
                           FinSi
              FinSi
FinSi
Fin
```

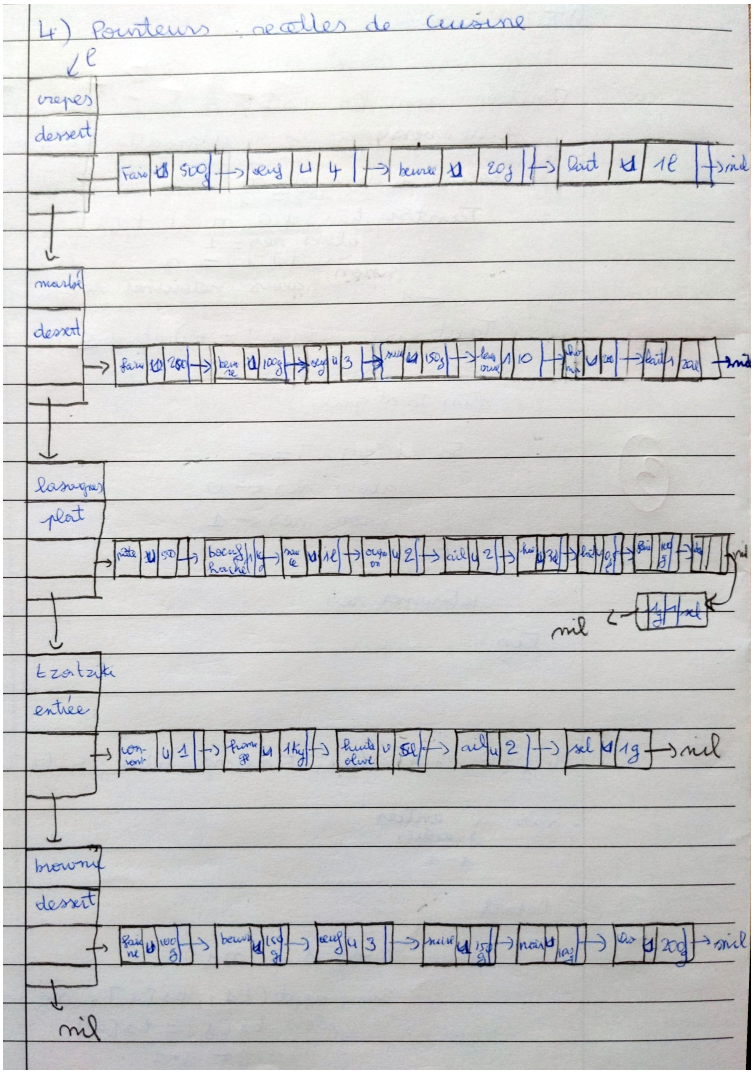
4. Pointeurs : recettes de cuisine – 9 pts

On souhaite représenter un livre de recettes de cuisine par une liste chaînée de recettes avec pour chacune la liste chaînée des ingrédients qu'elle utilise. Un ingrédient est donné par son nom `nomi` avec la quantité `qté` et l'unité `unité` utilisées (un ingrédient sera toujours donné dans la même unité). Une recette est donnée par son nom `nomr` et son type de plat `type` (« entrée », « plat » ou « dessert »).

```
Type  ingre = Enregistrement
                  nomi, unité : chaîne
                  qte : réel
                  suiv : ^ingre
              FinEnregistrement
      recette = Enregistrement
                  nomr, type : chaîne
                  pingre: ^ingre
                  suiv :^recette
                FinEnregistrement
      livre = ^recette
Var l : livre
```

4.1. Faire un dessin de la structure de données `l` avec les recettes suivantes (type et ingrédients entre parenthèses) : `u = unité`

- crêpes (dessert, farine 500 g, œuf 4 u, beurre 20 g, lait 1 l)
- marbré-chocolat (dessert, farine 250 g, beurre 100 g, œuf 3 u, sucre 150 g, levure 10 g, chocolat-noir 200 g, lait 20 cl)
- lasagnes (plat, pâtes-lasagnes 500 g, bœuf-haché 1 kg, sauce-tomate 1 l, oignon 2 u, ail 2 u, huile-olive 3 cl, lait 0.5 l, farine 100 g, beurre 10 g, sel 1 g)
- tzatziki (entrée, concombre 1 u, fromage-blanc 1 kg, huile-olive 5 cl, ail 2 u, sel 1 g)
- brownie-chocolat (dessert, farine 100 g, beurre 150 g, œuf 3 u, sucre 150 g, noix 100 g, chocolat-noir 200 g).



(Dessin tiré de la copie de Papa Thiecouta Diallo)

4.2. Ecrire en pseudo-langage une fonction qui crée la liste de toutes les recettes du livre `l` de type « dessert » qui contiennent du chocolat-noir.

```
Fonction donne-choco(l : livre) : livre
Var res, t : livre
    q : ^ingre
Début
res←nil
TantQue l≠nil Faire
   Si l^.type = 'dessert'
      Alors q←l^.pingre
            trouve←faux
            TanQue q≠nil et non trouve Faire
               Si q^.nomi='chocolat-noir'
                  Alors trouve←vrai
                        t←allouer(recette)
                        t^.nomr←l^.nomr
                        t^.type←l^.type
                        t^.pingre←nil
                        t^.suiv←res      {insertion en tête de res}
                        res←t
                  Sinon q←q^.suiv
               FinSi
```

```
            FinTantQue
   FinSi
   l← l^.suiv
FinTantQue
retourner(res)
Fin
```

4.3. Ecrire en pseudo-langage une fonction qui crée la liste des ingrédients et, pour chacun d'eux, les recettes qui le contiennent. **Définir un nouveau type de liste et faire un dessin.**

```
Type  ingre-bis = Enregistrement
                    nomi, unité : chaîne
                    precette : ^recette-bis
                    suiv : ^ingre-bis
                  FinEnregistrement
      recette-bis = Enregistrement
                      nomr, type : chaîne
                      suiv :^recette-bis
                    FinEnregistrement
      lingre = ^ingre-bis

Procedure ajoute-ingre(E q : ^ingre, l : ^recette, E/S lingre : lingre)
Var t, li : ^ingre-bis
    p,r : ^recette-bis
Debut
Si lingre=nil {si la liste est vide on ajoute l'ingrédient et la recette}
      Alors lingre←allouer(ingre-bis)
            lingre^.nomi←q^.nomi
            lingre^.unité←q^.unité
            lingre^.suiv←nil
            precette←allouer(recette-bis)
            r←precette
            r^.nomr←l^.nomr
            r^.type←l^.type
            r^.suiv←nil
      Sinon li←lingre
            TanQue li≠nil et li^.nomi≠q^.nomi Faire
               li←li^.suiv
            FinTantQue
            Si li=nil {on n'a pas trouvé l'ingrédient alors on l'ajoute}
               Alors t←allouer(ingre-bis)
                     t^.nomi←q^.nomi
                     t^.unité←q^.unité
                     t^.suiv←lingre
                     t^.precette←nil
                     lingre←t
                     li←lingre
            FinSi
            p←li^.precette
            TanQue p≠nil et p^.nomr≠li^.nomr Faire
               p←p^.suiv
            FinTantQue
            Si p=nil {on n'a pas trouvé la recette alors on l'ajoute}
               Alors r←allouer(recette-bis)
                     r^.nomr←l^.nomr
                     r^.type←l^.type
                     r^.suiv←li^.precette
                     li^.precette←r
            FinSi
            {Sinon la recette existe déjà pour l'ingrédient et on ne fait rien}
FinSi
Fin
```

```
Fonction donne-liste-ingre(l : livre) : lingre
Var res : lingre
    q : ^ingre
Début
res←nil
TantQue l≠nil Faire
   q←l^.pingre
   TanQue q≠nil et Faire
      ajoute-ingre(q,l,res) {si la recette existe déjà pour l'ingrédient,
                             l'ajout ne se fait pas}
      q←q^.suiv
   FinTantQue
   l← l^.suiv
FinTantQue
retourner(res)
Fin
```

4.4. Un menu est décrit dans un fichier texte dont chaque ligne correspond au nom d'une recette. Ecrire en pseudo-langage une procédure qui crée, à partir du fichier menu en entrée, un nouveau fichier texte avec la liste des courses à faire pour réaliser toutes ces recettes : chaque ligne est un ingrédient avec sa quantité et son unité (chaque ingrédient n'apparait qu'une fois dans le fichier). L'espace est le séparateur de mot.

Exemple :

fichier en entrée

```
tzatziki
lasagnes
crêpes
```

fichier en sortie

```
concombre 1 u
fromage-blanc 1 kg
huile-olive 8 cl
ail 4 u
sel 1 g
pâtes-lasagne 500 g
bœuf-haché 1 kg
sauce-tomate 1 l
oignon 2 u
lait 1.5 l
farine 600 g
beurre 30 g
œuf 4 u
```

```
Type ingredient = Enregistrement
                     nomi, unite : chaine
                     qte : réel
                  FinEnregistrement
      tabIngre = tableau[1..50] de ingredient


Procedure ajouter(E/S t : tabIngre, n : entier, E nom, u : chaine, q : réel)
Var i : entier
Début
i←1
TantQue t[i].nomi≠nom et i≤n Faire
      i←i+1
FinTantQue
Si t[i].nomi=nom
      Alors t[i].qte ← t[i].qte + q
      Sinon t[i].qte ← q
            t[i].nomi ← nom
            t[i].unite ← u
            n ← n+1
FinSi
Fin
```

```
Procedure créer-tab(E nom-menu : chaine , l : livre, S t : tabIngre, n : entier)
Var fmenu : FT
    r,c : chaine
    pi : ^ingre
Début
fmenu←OuvrirEnLecture(nom-menu)
TanQue non FinFichier(fmenu) Faire
      r←lireChaine(fmenu)
      TanQue l^.nomr≠r Faire {je suppose que ma recette appartient bien à l}
            l←l^.suiv
      FinTantQue
      pi←l^.pingre
      n←0
      TanQue pi≠nil Faire
            ajouter(t,n,pi^.nomi,pi^.unite,pi^.qte)
            pi←pi^.suiv
      FinTantQue
FinTantQue
fermer(fmenu)
Fin

Procedure créer-fich(E t : tabIngre, n : entier)
Var fliste : FT
    i : entier
    c : chaine
Début
fliste ← créerFichier('Liste-courses.txt')
Pour i←1 à n inc +1 Faire
     c←concatener(t[i].nomi,concatener(reel2chaine(t[i].qte),t[i].unite))
     écrireChaine(fliste,c)
FinPour
fermer(fliste)
Fin

Procedure créer-listeCourses(E nom-menu : chaine , l : livre)
Var t : tabIngre
    n : entier
Début
créer-tab(nom-menu,l,t,n)
créer-fich(t,n)
Fin
```