Algorithmes et Structures de Données
Mardi 22 Octobre 2013
Correction

1. Pile - 5 pts

Soient la fonction *z* définie comme suit pour *n* entier :

- si *n* est positif alors *z(n)=n*-1
- si *n* est négatif ou nul, *z(n)=z(z(n*+2*)*

1.1. Ecrire la fonction *z* en pseudo langage.

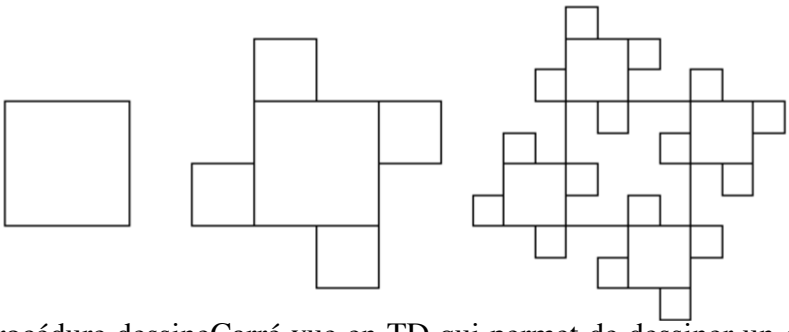
```
Fonction z(n : entier) : entier
Var res : entier
Début
Si n>0
      alors res←n-1
      sinon res←z(z(n+2){@1}){@2}
FinSi
Retourner(res)
Fin
```

1.2. Simuler la pile sur l'appel `écrire(z(-6)){@0}` dans le programme principal.

```
@2    n=1         res=0
@1    n=2         res=1
@2    n=0         res=0
@2    n=1         res=0
@1    n=2         res=1
@2    n=0         res=0
@2    n=1         res=0
@1    n=2         res=1
@2    n=0         res=0
@2    n=1         res=0
@1    n=2         res=1
@1    n=0         res=0
@1    n=-2        res=0
@1    n=-4        res=0
@0    n=-6        res=0
```

2. Récursivité - 5 pts

On souhaite dessiner la figure dont on a mis ci-dessous les premières étapes (profondeur 0,1 et 2).



On dispose de la procédure dessineCarré vue en TD qui permet de dessiner un carré dont on passe en paramètre les coordonnées de ses quatre points (le point en bas à gauche en premier puis en tournant dans le sens des éguilles d'une montre).

Ecrire la procédure `Carre(E xa,ya,d,f : entier)` où `xa` et `ya` sont les coordonnées du sommet en bas à gauche du grand carré, `d` est la longueur de son côté et `f` la longueur du côté du plus petit carré (on rappelle que l'origine de l'écran est placée en haut à gauche).

```
Procédure Carre(E xa,ya,d,f : entier)
Début
Si d>f
   Alors dessineCarre(xa,ya,xa,ya-d,xa+d,ya-d,xa+d,ya)
         Carre(xa-(d div 2),ya,d div 2,f)
         Carre(xa,ya-d,d div 2, f)
         Carre(xa+d,ya-(d div 2),d div 2, f)
```

```
        Carre(xa+(d div 2),ya+(d div 2), d div 2 ,f)
FinSi
Fin
```

3. Sous-séquences croissantes - 5pts

On souhaite écrire une procédure `déterminerSequences`, qui à partir d'un tableau d'entiers `t` de `n` éléments, affiche les sous-séquences strictement croissantes de `t`, ainsi que la sous-séquence la plus grande. On mémorisera les sous-séquences dans un tableau `Seq`, avec pour chacune d'elles son indice de début et sa taille.
Exemple :
Soit t un tableau de 15 éléments : 1 ; 2 ; 5 ; 3 ; 12 ; 25 ; 13 ; 8 ; 4 ; 7 ; 24 ; 28 ; 32 ; 11 ; 14
Les séquences strictement croissantes sont : < 1 ; 2 ; 5 >, < 3 ; 12 ; 25 >, < 13 >, < 8 >,
< 4 ; 7 ; 24 ; 28 ; 32 > et < 11 ; 14 > et la plus grande sous-séquence est < 4 ; 7 ; 24 ; 28 ; 32 >

3.1. Ecrire en pseudo-langage la déclaration du type du tableau `seq`.

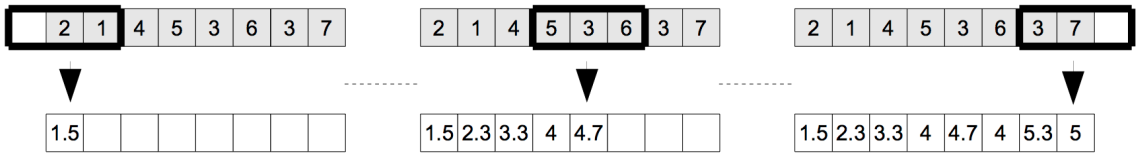
```
Const max = 50
Type tab-seq = tableau [1..max] de ss
     ss = Enregistrement
            ind, taille : entier
          FinEnregistrement
Var seq : tab
```

3.2. Ecrire en pseudo-langage la procédure `déterminerSequences`

```
Procédure déterminerSequences(E t:tab, n:entier ; S seq:tab-seq, m,k:entier)
Var i,j,k: entier
Debut
j←1
m←0
k←1
TantQue j<n Faire
      i←j
      m←m+1
      TantQue t[j]<t[j+1] et j<n Faire
            écrire(t[j])
            j←j+1
      FinTantQue
      écrire (‘↲’)
      seq[m].ind←i
      seq[m].taille←j-i+1
      Si t[m].taille>t[k].taille
            alors k←m
      FinSi
      j←j+1
FinTantQue
Si i=n
      Alors m←m+1
            seq[m].ind←i
            seq[m].taille←1
Finsi
écrire(k)
Fin
```

4. Lissage d'un tableau - 5pts

Ecrire en pseudo-langage une fonction `lisser (E t : tab ; n, k : entier ; S l : tab)` qui lisse les `n` valeurs réelles d'un tableau `t` dans un nouveau tableau `l` en utilisant une fenêtre glissante de taille `k` pour moyenner les valeurs de `t`. Pour les premières et dernières valeurs, seules les valeurs dans la fenêtre sont prises en compte.
Exemple : avec `t = 2 ; 1 ; 4 ; 5 ; 3 ; 6 ; 3 ; 7` et `k=3`




```
Procédure lisser (E t : tab ; n, k : entier ; S l : tab)
Var i : entier
Début
Pour i←1 à n inc +1 faire
      l[i]←calculer-moyenne(t,n,i,k)
FinPour
Fin

Fonction calculer-moyenne(t : tab ; n,i,k : entier) : réel
Var j,nb : entier
    s : réel
Début
s←0
nb←0
Pour j←i-(k div 2) à i+(k div 2) inc +1 faire
      Si j>=1 et j<=n
            alors s←s+t[j]
                  nb←nb+1
      FinSi
FinPour
Retourner(s/nb)
Fin
```