```
#define MAX 100
struct triplet {
  float x;
  float y;
  float z;
} ;
typedef struct triplet triplet;
typedef struct triplet vectri [MAX];
```

TABLE 1 – Type pour vecteur

Programmation Impérative

Partiel du 20/05/2016

Nom _____
Prénom _____
Login _____

Ce sujet doit être remis avec votre copie.

A) Fonction numérique

Commencez par calculer les premières valeurs de la suite suivante :

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= 3 \times f(n-1) - f(n-2) \end{aligned}$$

1) Donnez une fonction récursive qui renvoie le n-ième élément de la suite.
2) Donner une fonction itérative qui renvoie le n-ième élément de la suite.
3) Donnez une (en fait deux) fonction récursive terminale qui renvoie le n-ième élément de la suite.

B) Vecteurs

1) Soit la déclaration de la table 1, donnez une fonction d'un tel vecteur et de son nombre d'éléments, qui renvoie la somme des champs x du vecteur.
2) Donnez une fonction d'un tel vecteur et de son nombre d'éléments, qui pour chaque case du vecteur, remplit le champ z avec la valeur du champ x multipliée par celle du champ y.
3) Donnez maintenant une fonction qui renvoie la moyenne des champs z de ce vecteur.

C) Tris

Soit la déclaration de la table 2, il s'agit d'une structure composée d'un vecteur (les données) et un nombre de données.

```
struct vec {
  int nb_ele;
  double * v;
} ;
typedef struct vec vec_t;
```

TABLE 2 – Type de données

1) Donnez une fonction qui renvoie le plus grand élément du vecteur.
2) Donnez une fonction qui place le plus grand élément du vecteur à la fin de celui-ci en l'échangeant avec le dernier élément.
3) Donnez une fonction qui range le plus grand élément du vecteur dans un nouveau vecteur.
4) Utilisez ces fonctions pour créer une fonction qui range un vecteur dans l'ordre décroissant.

D) Arbres

Voici (en table 3) un type de données, `arbre`.

1. Faites une fonction qui affiche toutes les valeurs d'un tel arbre.
2. Faites tourner la fonction `donc` qui appelle calcul (voir table 4) avec comme paramètre la valeur 7040. Le contenu d'un certain nombre de cases mémoire est donné dans la table 5.
   Attention, pour faire tourner cette fonction, il faut, comme en cours, montrer ce qui arrive à chaque case définie.
3. Faites une fonction qui calcule la somme des champs *val* négatifs de l'arbre.
4. Faites une fonction qui calcule le nombre de noeuds internes de l'arbre (c'est un noeud qui a au moins un successeur).

E) Fichiers

1. Écrire une fonction qui place dans un fichier le nombre d'éléments puis tous les éléments d'un vecteur comme celui défini dans la table 2.
2. Faire une fonction qui permet de calculer combien d'éléments de type `double` sont écrits dans le fichier donné en argument.
3. Faire une fonction qui permet de sauvegarder dans un fichier tous les éléments d'un arbre (cf. table 3) donné en argument.

```
struct noeud {
  int num;
  double val;
  struct noeud * un;
  struct noeud * deux;
  struct noeud * trois;
} ;
typedef struct noeud noeud_t;
typedef struct noeud * arbre;
```

TABLE 3 – Types arbres

```
struct paire {
  int n;
  double v;
} ;
typedef struct paire paire_t;
paire_t calcul (arbre a) {
  paire_t p, q;
  if (! a) {
p.n = 0;
p.v = 0.0;
return p;
  }
  p.n = a->num;
  p.v = a->val;
  q = calcul (a->un);
  p.n += q.n;
  p.v += q.v;
  q = calcul (a->deux);
  p.n += q.n;
  p.v += q.v;
  q = calcul (a->trois);
  p.n += q.n;
  p.v += q.v;
  return p;
}
double donc (arbre a) {
  paire_t p;
  p = calcul (a);
  if (p.n)
return p.v / p.n;
  return 0.0;
}
```

TABLE 4 – Fonction donc

```
7040 1
7044 0.50
7048 7184
7056 0
7064 7088

7088 2
7092 1.10
7096 7232
7104 0
7112 7136

7136 3
7140 1.70
7144 7296
7152 0
7160 0

7184 4
7188 0.30
7192 7344
7200 0
7208 0

7232 5
7236 0.90
7240 7392
7248 0
7256 0

7296 6
7300 1.50
7304 0
7312 0
7320 0

7344 7
7348 0.10
7352 0
7360 0
7368 0

7392 8
7396 0.70
7400 0
7408 0
7416 0
```

TABLE 5 – Mémoire