Programmation Impérative

18/2/2025

Nom _____
Prénom _____
numéro _____
mail _____

Les questions sont à traiter dans l'ordre qui vous plaira mais vous n'aurez peut-être pas le temps de tout faire...

Ce sujet doit être remis avec votre copie, une version imprimable sera donnée sur la page du cours pour que vous puissiez retravailler les questions.

Quand vous simulez le fonctionnement de l'ordinateur lors de l'exécution d'une fonction, vous devez donner l'adresse et les contenus successifs de la mémoire pour chacune des variables utilisées. Pour les fonctions récursives tous les appels doivent être spécifiés.

A) Simples

1. Écrire une fonction itérative d'un argument x, qui renvoie la plus petite valeur de la suite de Fibonacci supérieure à x.
   Exemple : petitfib(6) renvoie 8.
2. Soit la fonction `nope`, cf. Table 1, donnez toutes les étapes de l'exécution de cette fonction.
3. Soit la structure de vecteur donnée en Table 2, donnez une fonction qui renvoie la somme des nombres négatifs de ce vecteur.
4. Soit la structure de vecteur donnée en Table 2, donnez une fonction d'une telle structure et d'un nombre x qui renvoie le nombre de fois que cette valeur est dans le vecteur.
5. Soit la fonction g, donnez une forme récursive de cette fonction.

$$\begin{aligned} g(0) &= 0 \\ g(1) &= 1 \\ g(n) &= g(n-1) + g(n-2) + 2 \end{aligned}$$

6. Donnez une forme itérative de cette fonction g.

```
void hope (float * a, float * b) {
  float c;
  c = *a;
  while (c != *b)  {
    if (c > *b) {
      c -= *b;
    }
    else {
      *b -= c;
    }
  }
  *a = c;
}
void nope () {
  float x, y;
  x = 120.0;
  y = 45.0;
  hope (&x, &y);
  printf(" x == %f\n", x);
  x = 125.0;
  y = 21.0;
  hope (&x, &y);
  printf(" x == %f\n", x);
}
```

Table 1 – Fonctions `hope` et `nope`

```
typedef struct vecfl {
  int nbele;
  float * f;
} vecfl_t;
```

Table 2 – Structure

```
int lav (int a, int b) {
  if ( b == 0)
    return -1;
  if (a < b)
    return 0;
  return lav (a-b, b) + 1;
}
```

Table 3 – Fonction lav

```
typedef unsigned char uchar;
typedef unsigned int uint;
typedef struct binaire {
  int nbent;
  int nbfrac;
  uchar *ent;
  uchar *frac;
} binaire_t;
typedef struct decimal {
  int nbent;
  int nbfrac;
  uchar *ent;
  uchar *frac;
} decimal_t;
```

TABLE 4 – Deux structures du cours

7. Écrivez une fonction itérative qui renvoie le reste de la division de ses deux arguments.
8. Vous trouverez en table 3 une fonction `lav`, donnez sa forme itérative.

B) Binaire et décimal

Nous avons travaillé sur deux structures de nombres binaires et décimaux, vous les trouverez dans la table 4.

1. Écrivez une fonction qui prend deux nombres décimaux ainsi définis et renvoie 1 s'ils sont égaux et 0 sinon.
2. Écrivez une fonction qui prend deux nombres binaires ainsi définis et renvoie le plus grand des deux.
3. Donnez une fonction qui prend un décimal ainsi défini et qui lui rajoute 1.
4. Donnez une fonction qui prend un décimal ainsi défini et qui le convertit en un nombre de type `double`.

C) Vecteurs

Nous définissons une structure de vecteur pour les binaires. Nous avons vu en cours une fonction permettant d'afficher un tel nombre,

```
void aff_bin (binaire_t ) ;
```

1. Écrire une fonction permettant d'afficher tous les éléments d'un tel vecteur.

```
typedef struct vecbin {
  int nbele;
  binaire_t * vec;
} vecbin_t;
```

TABLE 5 – Vecteur de binaires

2. Écrire une fonction qui renvoie le plus grand nombre du vecteur.
3. Écrire une fonction qui fait le miroir du vecteur (le premier élément deviendra le dernier, le second l'avant-dernier,...).