Partiel Programmation

06 fevrier 2024

Aucun document électronique n'est autorisé

Durée 3h

Exercice 1 ✓ a) Ecrire une fonction qui détermine si le tableau T1 est un préfixe du tableau T2.

Dans un premier temps donner la signature de la fonction (nom, type de retour et paramètres en nombre et en type) ainsi que les contraintes (Préconditions et Postconditions) exprimées en langage naturel. Exemple de préfixe :

. T1 : [1, 2, 4, 3], T2 : [1, 2, 4, 3] c'est vrai

. T1 : [1, 4, 4, 3], T2 : [1, 2, 4, 3] c'est faux

. T1 : [1, 2], T2 : [1, 2, 4, 3] c'est vrai

✓ b) Ecrire une fonction qui détermine si le tableau T1 est un suffixe du tableau T2. Exemple :

. T1 : [4, 3], T2 : [1, 2, 4, 3] c'est vrai

. T1 : [1, 4, 3, 4], T2 : [1, 2, 4, 3] c'est faux

c) Ecrire une fonction qui calcul le ppcm (plus petit commun multiple) de deux entiers positifs.

Donner d'abord la signature de la fonction ppcm ainsi que ses contraintes. Exemple, ppcm de 15 et 23 est 345, un autre exemple le ppcm de $x = 15$ et $y = 27$ est 135. Comment procéder ? on décompose chaque entier en produit de facteurs premiers, par exemple $27 = 3 * 9$, mais l'entier 9 n'est pas premier on le décompose, $9 = 3 * 3$ le résultat est $27 = 3 * 3 * 3$ ensuite on met le résultat sous sa forme de puissance (si possible). Ainsi $27 = 3^3$. Ensuite on décompose l'entier 15 en facteurs premiers (toujours). Par exemple $15 = 3 * 5$. Il n'y a pas de possibilité de regroupement sous forme de puissance, on le garde sous sa forme de produit de facteurs premiers. le résultat est si il ya deux facteurs identiques dans x et dans y on garde le facteurs du plus grande puissance. Si il ya des facteurs différents dans les deux nombres on les garde tous. Ainsi pour $x = 15 = 3 * 5$ et $y = 27 = 3^3$ on effectue le produit suivant : $3^3 * 5 = 135$. Le ppcm de x et y est égal à 135. Vous pouvez utiliser votre propre méthode de résolution du ppcm (il ya plusieurs approches).

Exercice 2 Soit f un entier positif. On définit la suite u_n par :

$$\begin{cases} u_0 = f \\ u_{n+1} = u_n + \text{produit des chiffres de } u_n \end{cases}$$

✓ a) Ecrire une fonction `int produit(int n)` qui retourne le produit des chiffres de son paramètre entier n . Par exemple, `produit(244) = 2 * 4 * 4 = 32`. Cette fonction ne fait aucun affichage à l'écran. Nous rappelons que la division entière en langage C s'écrit en utilisant l'opérateur `/` et le reste de la division entière s'écrit au moyen de l'opérateur `%`.

✓ b) Ecrire une fonction `int u(int f, int n)` qui retourne le n -ième terme de la suite u . Par exemple pour : `u(244, 1) = 276`, c'est à dire $u_1 = 276$ avec $u_0 = f = 244$

$u(244, 2) = 360$, c'est à dire $u_2 = 360$ avec $u_0 = f = 244$

$u(244, 5) = 360$, c'est à dire $u_5 = 360$ avec $u_0 = f = 244$

- c) Ecrire une fonction `int fecondite(int f)` qui renvoie la fécondité de l'entier f . Cette fonction utilisera la fonction `int u(int f, int n)` de la question précédente.
- d) Ecrire une fonction récursive `int fecondite_recursive(int f)` qui renvoie la fécondité de l'entier f . Cette fonction n'utilisera **pas** la fonction `int u(int f, int n)` de la question b.
- e) Ecrire un programme principal qui boucle en demandant un entier f à l'utilisateur. Tant que l'entier saisi est positif le programme calcule, affiche sa fécondité et la suite correspondant à sa fécondité, puis redemande un nouvel entier. Dès que l'entier saisi est négatif, le programme s'arrête.

✓ **Exercice 3** Initialiser le tableau `tab` de la fonction `main` ci-dessous, avec 9 entiers satisfaisant aux exigences du problème. Ensuite, comprendre le programme qui suit, en donnant le résultat final, accompagné d'un schéma explicatif.

```
//-----
#include<stdio.h>
//-----
void echange(int t[], int i, int j){
    int tmp = t[i];
    t[i] = t[j];
    t[j] = tmp;
}
//-----
void FX(int t[], int T){
    int b, w, r;

    b = w = 0; r = T-1;
    while (w <= r ){
        if(t[w] == 2)
            w++;
        else{
            if(t[w] == 1){
                echange(t, b, w);
                b++;
                w++;
            }
            else{
                while(t[r] == 3 && w < r)
                    r--;
                echange(t, r, w);
                r--;
            }
        }
    }
}
//-----
```

```

main(int argc, char** argv){
    int tab[] = {3,...}; /* <===== Initialiser et completer*/
int i;
    FX(t, 9);
    for(i = 0; i < 9; i++)
        printf("%d ",tab[i]);
    printf("\n");
return 0;
}
//-----

```

✓ **Exercice 4** Les exercices suivants n'utiliseront que des pointeurs (aucun index et on n'utilisera aucune fonction de la librairie C).

- Ecrire une fonction qui affiche les éléments du tableau, vecteur, ci-dessous.
- Ecrire une fonction qui donne la taille en nombres de caractères de chacune des chaînes (ligne) de caractères du tableau ci-dessous.
- Quel est le nombre total de caractères du tableau ci-dessous ?
- Ecrire une fonction qui donne le nombre de chaînes de caractères identiques du tableau ci-dessous.
- Ecrire une fonction qui donne le nombre de mots du tableau. Un mot est séparé d'un autre par le caractère espace ' '.
- Compléter le programme suivant pour tester les fonctions précédentes.

```

int main(){
char * vecteur[] = {
    "Rien n'est impossible en algorithmique",
    "Il n'y a pas de probleme",
    "qui ne soit sans solution.",
    "on insiste",
    "Il n'y a pas de probleme",
    "qui ne soit sans solution.",
    "Donc s'il n'y a pas de solution,",
    "c'est qu'il n'y a pas de probleme !"
};
//TODO
}
//-----

```

- Dans un fichier texte, nouvellement créé, vous insérez dans la première ligne de ce fichier le nombre de caractère suivi par le nombre de ligne du texte ci-dessus (contenu dans le tableau de nom vecteur). Ensuite dans le même fichier vous copiez (après la première ligne) les lignes du tableau, vecteur, où chaque ligne est précédée par un numéro de ligne en commençant par 0 (zéro). Exemple :
 - c caractères l lignes,
 - 0 "Rien n'est impossible en algorithmique",
 - 1 "Il n'y a pas de probleme",
 - ...