I2010 : langage C (TP7)

Fonctions

1. Spécifications de fonctions

1. a. Ecrivez les spécifications de la fonction suivante :

```
void imprimer (const char *s , char **t) {
    if (t == NULL){
         printf("table absente\n");
         return;
    }
    if (*t == NULL){
         printf("table vide\n");
         return;
    }
    printf("%s\n", s);
    for (char **p = t; *p != NULL; p++) {
         printf("%s\n", *p);
    }
}
```

Réponse :

```
/* Affiche un tableau de chaînes
 * PRE: s: chaîne de caractères
 *      t: table de chaînes terminé par NULL
 * POST: affiche la chaîne s puis les chaînes de t;
 *       affiche un message spécifique si t est NULL ou vide
 */
```

1. b. Ecrivez les spécifications de la fonction suivante :

```
void imprimer (const char *s , int *t, int taille) {
    if (t == NULL){
         printf("table absente\n");
         return;
    }
    if (taille == 0){
         printf("table vide\n");
         return;
    }
    printf("%s\n", s);
    for (int i=0; i<taille; i++) {
         printf("%i\n", t[i]);
    }
}
```

Réponse :

```
/* Affiche un tableau d'entiers
 * PRE: s: chaîne de caractères
 *      t: table de taille entiers
 * POST: affiche la chaîne s puis les entiers de t;
 *       affiche un message spécifique si t est NULL ou vide
 */
```

1. c. Ecrivez les spécifications de la fonction suivante :

```
int rechercher (const char *s, char c) {
    if (s == NULL){
         return -1;
    }
    for (char *p = s; *p != '\0'; p++) {
         if (*p == c)
             return p-s;
    }
    return -1;
}
```

Réponse :

```
/* Recherche un caractère dans une chaîne
 * PRE: s: chaîne de caractères
 *      c: caractère à rechercher dans s
 * RES: renvoie l'indice de la première occurrence de c
 *      dans la chaîne s ; -1 si c n'a pas été trouvé
 */
```

2. Prototypes de fonctions

2. a. Ecrivez le prototype d'une fonction qui reçoit en argument le chemin complet d'un fichier. Elle renvoie vrai si un tableau passé en paramètre a été rempli avec les lignes du fichier converties en entiers et ses tailles physique et logique mises à jour ; faux sinon.
Nommez les paramètres de façon explicite.

Réponse :

```
bool chargerEntiers (const char* filename, int **tab, int *tailleP, int *tailleL);
```

Dans ce prototype et les suivants, l'opérateur étoile en orange * correspond à une indirection introduite pour pouvoir modifier le paramètre qui est, pour rappel, toujours passé par valeur en C. Fournir l'adresse d'un paramètre permet à la fonction de modifier sa valeur.
A priori, à l'intérieur de votre fonction, `tab`, `tailleP` et `tailleL` n'apparaîtront jamais directement car il s'agit en réalité des adresses des paramètres réels ; par conséquent, le corps de votre fonction utilisera plutôt : `(*tab)`, `(*tailleP)` et `(*tailleL)`.

2. b. Ecrivez le prototype d'une fonction qui reçoit en argument le chemin complet d'un fichier. Elle renvoie vrai si un tableau passé en paramètre a été rempli avec les mots du fichier et ses tailles physique et logique mises à jour ; faux sinon.
Nommez les paramètres de façon explicite.

Réponse :

```
bool chargerMots (const char* filename, char ***tab, int *tailleP, int *tailleL);
```

2. c. Ecrivez le prototype d'une fonction qui sépare les mots d'une chaîne de caractères. Elle les sauve dans un tableau passé en paramètre, sauve sa taille physique fournie en paramètre et renvoie sa taille logique. Elle renvoie -1 si une erreur s'est produite.
Nommez les paramètres de façon explicite.

Réponse :

```
int decouper (const char* s, char*** t, int* tailleP);
```

3. *Implémentation de fonctions*

3. a. Ecrivez une fonction qui permute la valeur de deux entiers.

Réponse :

```
void permuter (int *a, int *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

3. b. Ecrivez une fonction qui permute le contenu de deux chaînes de caractères.

Réponse :

```
void permuter (char **s1, char **s2) {
    char* tmp = *s1;
    *s1 = *s2;
    *s2 = tmp;
}
```

3. c. Ecrivez une fonction qui permute le contenu de deux tableaux d'entiers.

Réponse :

```
void permuter (int **t1, int **t2) {
    int* tmp = *t1;
    *t1 = *t2;
    *t2 = tmp;
}
```

Note : remarquez que vous ne pourrez pas tester cette fonction avec des tableaux statiques. En effet, un tableau statique tel que `int tab[4]` ne désigne pas en lui-même un pointeur mais bien un *tableau*, de type *tableau de 4 entiers*[1]. Vous ne pourrez donc passer l'adresse d'un tel tableau à votre fonction.

4. *Débogage de fonction*

Expliquez le(s) problème(s) que présente le code suivant et proposez une solution, sans modifer le prototype de la fonction :

```
/* Crée un tableau avec les puissances d'un nombre
 * PRE: n: nombre entier strictement positif
 * RES: renvoie un tableau de taille n+1
 *      rempli avec les puissances de n
 */
int* creerTable (int n) {
   int t[n+1];
   t[0] = 1;
   for (int i=1; i<=n; i++)
      t[i] = t[i-1] * n;
   return (int*)t;
}
```

[1] https://fr.wikibooks.org/wiki/Programmation_C/Tableaux#Exceptions_%C3%A0_la_r%C3%A8gle_de_conversion

Réponse :

`t` est une variable locale qui sera détruite au retour de la fonction ; de plus, la tableaux de taille variable (*variable length array*) sont interdits en C ➔ la mémoire doit être allouée dynamiquement :

```
int* creerTable (int n) {
    int *t = (int*) malloc((n+1) * sizeof(int));
    if (t != NULL) {
       t[0] = 1;
       for (int i=1; i<=n; i++)
          t[i] = t[i-1] * n;
    }
    return t;
}
```

5. *Exercice récapitulatif*

Réponse :

Voir ***ex7.5_solution.c*** ainsi que la vidéo explicative

➔ Voir **`ex7.5_solution.c`** ctions suivantes (les specs sont précisées dans le fichier source)