ENSEIRB-MATMECA – Filière Electronique

Examen Langage C - PG108
2019-2020 2ème session
Jeudi 12 mars 2020
Durée de l'épreuve : 2h / documents autorisés – sans calculatrice

1) Fonctions mystère:

Que font les fonctions suivantes ?
Pour chaque fonction, maximum : une phrase pour expliquer ce que fait la fonction, éventuellement une phrase supplémentaire par paramètre pour en expliquer l'utilité.

```
int laurel(int hardy) {
     if ((hardy % 2) == 0)
          return 0
     return 1;
}
```

```
int pim(char *pam) {
     for (int poum=0; poum < 10; poum++)
          if (pam[poum] == 0)
               return 0;
     return 1;
}
```

```
double john(double *paul, int george) {
     double ringo = *paul;
     while (--george > 0)
          if (ringo > paul[george])
               ringo = paul[george];
     return ringo;
}
```

```
void minus(char *cortex) {
     while (*cortex) {
          if ((*cortex >= 'A') && (*cortex <= 'Z'))
               *cortex += 'a' - 'A';
          cortex++;
     }
}
```

2) Exercice:

Écrivez un programme complet comprenant les inclusions de bibliothèque, la fonction *main* et éventuellement les sous-fonctions. Ce programme récupère une chaîne de caractères (*cc*) et un entier (*n*) qui seront fournis comme arguments sur la ligne de commande (dans cet ordre). Le programme affiche la fin de la chaîne *cc* en commençant au $n^{ième}$ caractère, puis affiche le début de *cc* en s'arrêtant au $(n-1)^{ième}$ caractère.

3) Le *jeu* des 3 erreurs:

La fonction suivante compte le nombre de valeurs négatives dans un tableau de doubles. 3 erreurs se sont glissées dedans.
- Mentionnez chacune de ces trois erreurs
- Re-écrivez la fonction en les corrigeant

```
int count_negs(double *tableau, int longueur) {
     float negs = 0;
     for (indice = 0;  indice <= longueur; indice ++)
          if (tableau[indice] < 0)
               ++negs;
     return negs;
}
```

4) Problème:

Les questions du problème peuvent être traitées de manière indépendante. Prenez le temps de lire le sujet jusqu'au bout.

Dans ce problème, nous allons nous intéresser à l'écriture d'une fonction capable d'effectuer du remplacement de texte. Cette fonction (que nous appellerons *replace*) prend trois chaînes de caractères en argument :
- *text* : le texte sur lequel il faut faire les opérations de remplacement
- *pattern* : le texte à remplacer
- *new_pattern* : le texte qu'il faut mettre à la place de *pattern* dans *text.*

Le résultat de la fonction *replace* devrait être une chaîne de caractères.

a) Pour rappel, une chaîne de caractères est implémentée dans un tableau, or, une fonction en C ne peut pas renvoyer de tableau. Quel mécanisme permet de faire en sorte que la fonction renvoie quelque chose qui se comportera comme une chaîne de caractères ?
Dans la suite du sujet, par abus de langage, nous considérerons que la fonction *replace* renvoie une chaîne de caractères.

b) Donnez le prototype de la fonction *replace.*

c) Écrivez la fonction *match* qui, à partir de deux chaînes (*text* et *pattern*) passées en argument, renvoie 1 si la chaîne la plus longue des deux commence comme la chaîne la plus courte, ou 0 sinon (si les premiers caractères sont différents deux à deux).

d) Écrivez la fonction *count*, qui reçoit deux chaînes (*text* et *pattern*) et qui renvoie le nombre de fois où *pattern* est présent dans *text*.
NOTE : La question *d)* arrivant après la question *c)*, la fonction *match* pourrait avoir une utilité, vous avez le droit de l'utiliser, même si vous n'avez pas répondu à la question *c)*.

e) Connaissant le nombre de fois ou la chaîne *pattern* apparaît dans *text*, quelle est la taille minimale du tableau qui contiendra le résultat à construire ?

f) Écrivez la fonction *len* qui renvoie la longueur d'une chaîne de caractères qu'elle reçoit en argument.

g) Écrivez la ligne qui permet à une fonction d'obtenir/créer un tableau dont la taille est *longueur*, tout en garantissant que le tableau ne soit pas détruit à la fin de la fonction (*longueur* est une variable et sa valeur n'est pas connue au moment de la compilation).

h) L'algorithme de la fonction consiste maintenant à tester caractère par caractère si ce caractère est le début d'un motif à remplacer. Si c'est le cas, il faut copier le nouveau texte et sauter la taille du motif dans le texte d'origine, sinon, il faut copier le caractère et passer au suivant.

Compte tenu des réponses aux questions précédentes, écrivez une première version de la fonction *replace* complète. Si vous n'avez pas répondu à une question précédente, vous pouvez simplement indiquer que vous en utilisez le résultat, même si vous ne le connaissez pas.

i) La version précédente est peu optimale car elle teste deux fois la présence du motif pour chaque caractère (la fonction *match* est appelée une première fois pour compter le nombre de remplacements, puis une seconde fois pour réellement effectuer le remplacement). Modifiez la fonction *count* de la question *d)* pour mémoriser dans un tableau les résultats des premiers tests (chaque case du tableau correspondant à un caractère dans le chaîne *text*). Il suffira ensuite de consulter ce tableau pour éviter de réitérer l'appel à la fonction *match*.

j) Quelle est, en octets, la quantité de mémoire utilisée par cette nouvelle version de *count ?*

k) Écrire une deuxième version de la fonction *replace*.

l) Proposez une méthode pour une troisième version de la fonction *replace* qui utilise moins de mémoire, sans pour autant augmenter le nombre d'appels à la fonction *match*. (L'écriture de cette troisième version n'est pas demandée, mais peut apporter des points hors barème)