Liavona ZHELTANOSAVA
Mark: 11.5/20 (total score: 11.5/20)

+15/1/50+

UNIVERSITÉ NICE SOPHIA ANTIPOLIS
POLYTECH NICE-SOPHIA

2015-2016

Nom : Zheltanosava **Groupe:** 2

Prénom : Liavona

Programmation Procédurale – SI3

Mardi 20 octobre 2015

Documents non autorisés. **Durée: 1h30**

Vous pouvez omettre les directives `#include` dans vos réponses. Par ailleurs, vous apporterez un très grand soin à ***la présentation*** *car elle* ***interviendra dans la notation.***

Question 1: Écrire la fonction `void shuffle(char *s)` qui mélange les caractères de la chaîne `s` de la façon suivante: on tire au sort le premier caractère et on le met dans `s[0]`, on tire au sort ensuite le deuxième caractère que l'on met dans `s[1]`, ...

On pourra utiliser la fonction `int rand(void);` qui renvoie un nombre aléatoire compris entre 0 et `RAND_MAX` (un nombre très grand). On utilisera le moins de tableaux locaux que possible, éventuellement aucun.

3/4

☐0 ☐1 ☐2 ☒3 ☐4 ☐.5 *Réservé au correcteur*

```
void shuffle(char *s){
    char tmp;
    int index_tmp = rand();
    int size = strlen(s);
    for(int i=0; i<size; i++){
        tmp = s[i];
        while(index_tmp > size-1){
            index_tmp = rand();
        }
        s[i] = s[index_tmp];
        s[index_tmp] = tmp;
    }
}
```

Bien bellement très grand donc ça peut prendre du temps.
Un simple modulo suffit.

Question 2: Un programme C délimite les blocs avec des accolades. On veut réaliser un analyseur simple qui indique si les accolades sont bien positionnées dans un programme, c'est-à-dire si on a bien autant d'accolades ouvrantes que fermantes. Attention, un simple comptage ne suffit pas: un programme qui comporterait 3 accolades fermantes suivies de 3 accolades ouvrantes n'est pas correct.

Votre programme prendra son entrée sur le fichier standard d'entrée et produira sa réponse sur le fichier standard de sortie. On suppose ici qu'il n'y a pas d'accolades dans les commentaires ou les chaînes du programme.

1.5/4

☐0 ☒1 ☐2 ☐3 ☐4 ☒.5 *Réservé au correcteur*

```
int accolades () {
    char ac[BIG_NUMBER];  // BIG_NUMBER est très grand
    int index = 0;
    int ac_ouvr = 0;
    int ac_fer = 0;
    char c;
    while ((c = getchar()) != EOF) {
        if (c == '{') {
            ac[index] = c;
            index++;
            ac_ouvr++;
        }
        if (c == '}') {
            ac[index] = c;
            index++;
            ac_fer++;
        }
    int size = strlen(ac);
    if (ac[0] == '}' || ac[size-1] == '{' ||
        ac_ouvr != ac_fer)
        return 0;
    return 1;
}
```

inutile ??

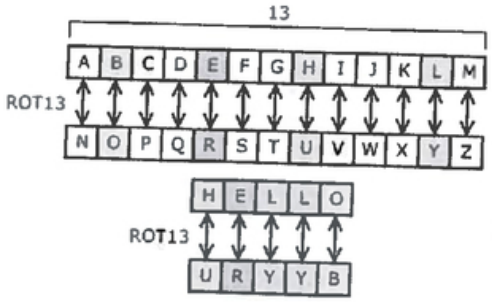
c'est bon

il faudrait mettre 1 seul accolade

et au milieu ??

si la 1° accolade est une ac_fermante ou
si la dernière accolade est une ac_ouvrante
ou si les nbs des ac_ouvr et des ac_ferm sont
différent alors on renvoie 0
sinon, c'est bon et on renvoie 1

Question 3: **Rot13** (*rotate by 13 places*) est un algorithme simpliste de chiffrement de texte, de type César. Comme son nom l'indique, il s'agit d'un décalage de 13 caractères de chaque lettre du texte à chiffrer. Son principal aspect pratique est que le codage et le décodage se font exactement de la même manière (source *Wikipedia*). Comme on le voit sur la figure suivante, chiffré avec **ROT13**, le mot «HELLO» devient «URYYB» (et inversement).



Écrire la fonction `void rot13(char message[]);` qui applique le codage **rot13** aux majuscules et minuscules de la chaîne de caractère `message` et laisse les autres caractères inchangés.

```
char msg[] = "Hello, World!";
rot13(msg); printf("=> %s", msg); // Affiche "=> Uryyb, Jbeyq!"
rot13(msg); printf("=> %s", msg); // Affiche "=> Hello, World!"
```

La fonction `rot13` **n'utilisera pas** de tableau de caractères intermédiaire.

3.5/4

☐0 ☐1 ☐2 ☒3 ☐4 ☒.5 *Réservé au correcteur*

```
void rot13(char message[]) {
    for(int i = 0; i < strlen(message); i++) {
        if(('a' <= message[i] && message[i] <= 'm') ||
           ('A' <= message[i] && message[i] <= 'M')) {
            message[i] += 13;
        }
        if(('n' <= message[i] && message[i] <= 'z') ||
           ('N' <= message[i] && message[i] <= 'Z')) {
            message[i] -= 13;
        }
    }
}
```

nan $O(n^2)$

Question 4: On a un programme constitué des fichiers suivants: `A.c`, `A.h`, `B.c`, `B.h`, `C.c`, `C.h`, `D.c`, `D.h` et `commun.h`. Le fichier `commun.h` contient des déclarations communes et est inclus par tous les fichiers `'.c'`. Tous les fichiers `'.c'` incluent le `'.h'` qui leur correspond. Enfin les fichiers `B.c` et `C.c` utilisent des données définies dans les fichiers `A.h` et `D.h`. Ce programme est ecrit en C99 et doit être compilé par `gcc` avec les options `-Wall -std=c99`.

Ecrire un fichier `Makefile` permettant de "gérer" ce programme. Vous pouvez/devez utiliser ici le plus possible les règles implicites connues de la commande **make**.

0/4

☒0 ☐1 ☐2 ☐3 ☐4 ☐.5 *Réservé au correcteur*

Question 5: On appelle nombre de Harshad, un nombre entier divisible par la somme de ses chiffres. Par exemple 12 (=3*4), 18, 20, 21... sont de tels nombres. Ecrivez la fonction harshad qui, étant donné un entier, détermine s'il s'agit d'un nombre de Harshad ou pas.

3.5/4

☐0 ☐1 ☐2 ☒3 ☐4 ☒.5 *Réservé au correcteur*

```
int harshad (int n) {
    int sum = 0;
    int tmp = n;
    if (n == 0) return 0;
    if (n < 10) return 1;
    while (n > 10) {
        sum += n % 10;   // on ajoute le reste
        n = n / 10;
    }
    sum += n;
    if ((tmp % sum) == 0) return 1;
    return 0;
}
// 0 signifie que n n'est pas un nombre de Harshad
// 1 signifie que n est un nb de Harshad
```

Ce n'est pas 1 cas particulier en fait

Compliqué

// pour n = 0 on pourra faire return 1 si 0 est considéré comme un nombre de Harshad mais apparemment il n'y est pas