

### Exercice I: Chaîne mystérieuse (6 points)

Dessiner l'état de la mémoire traçant le programme suivant. Déduire la sortie.

```
#include <stdio.h>
char* f2(char *s){
    if (*s=='\0' || *(s+1)=='\0' || *(s+2)=='\0')
        return s;
    *s+=11;
    return f2(++s);
}
char* f1(int n, char* str){
    str+=n;
    f2(str);
    *(str+1)-=3;
    return str;
}
void test(){
    int n=3;
    char str[]="difficult or easy";
    printf("%s", f1(n++, str+10));
}
void main() {
    test();
}
```

### Exercice II: Don du sang par type compatible (10 points)

Les types de sang sont très importants lorsqu'une transfusion sanguine est nécessaire. Dans une transfusion sanguine, un patient doit recevoir un groupe sanguin compatible avec son propre groupe sanguin. Si les groupes sanguins ne sont pas compatibles, cela peut entraîner la mort. Par conséquent, il est important que les groupes sanguins correspondent les uns aux autres avant les transfusions sanguines. Pour simplifier, le tableau suivant montre 3 types de sang différents, ainsi que leurs types de donneurs et de receveurs correspondants.

		Receveur		
		A	B	C
Donneur	A	x		
	B	x	x	
	C	x	x	x

Modified version of a blood type chart

Nous voulons garder une trace des donneurs disponibles avec leurs groupes sanguins dans une liste chaînée de donneurs. Chaque donneur est identifié par: son nom, son numéro de téléphone et son groupe sanguin.

En cas d'urgence, un donneur compatible sera contacté pour une transfusion sanguine.

1. Définir les types **Donneur** et **Noeud**. Le type Noeud représente un nœud dans une liste linéaire chaînée de donneurs.

2. Ecrire la fonction `trouve1Donneur` qui étant donné le groupe sanguin du receveur et la liste des donneurs, renvoie un pointeur sur la première occurrence d'un donneur avec un groupe sanguin compatible si ceci existe.

Le technicien de laboratoire n'était pas satisfait de la structure de données proposée. Il s'est plaint du fait qu'il faut trop de temps pour rechercher un donneur compatible. Nous avons donc restructuré les données ci-dessus en 3 listes chaînées afin de séparer les groupes sanguins disponibles. Nous avons utilisé la déclaration suivante pour stocker les 3 listes chaînées:

```
Noeud* donneurs[3];
```

3. Réécrire la fonction `trouve1Donneur` en prenant en tant que paramètres le groupe sanguin du receveur et le tableau des donneurs.

### Exercice III: Boucle ça (6 points)

Ecrire une fonction récursive `loopIt` qui modifie une liste linéaire simplement chaînée en une liste circulaire. Dans une liste linéaire circulaire, le dernier nœud pointe vers le premier nœud de la liste.

*Bonne chance!*