Département Informatique

Nom :		Année scolaire :	2006–2007
Prénom :		Date :	13 janvier 2007

Module INF423
Session de janvier

—

Programmation avancée en C

—

Contrôle de connaissance[1] de 45 minutes

ERCI de répondre (au moins) dans les blancs.
Lire tout le sujet en entier du début à la fin, en commençant à la première page et jusqu'à la dernière page, avant de commencer à répondre : cela peut vous donner de l'inspiration et vous permettre de mieux allouer votre temps en fonction de vos compétences.

Chaque question sera notée entre 0 et 10 et la note globale sera calculée par une fonction des notes élémentaires. La fonction définitive sera choisie après correction des copies.
Attention : tout ce que vous écrirez sur cette copie pourra être retenu contre vous, voire avoir une influence sur la note d'INF423.

1 Généralités

Question 1 : Quelle valeur le programme suivant affiche-t-il lors de l'exécution ? (Durée ≈ 1 minute)

```
1 #include <stdio.h>
  #include <stdlib.h>
3
  int *fonction(int valeur) {
5     valeur = valeur + 3;
```

[1] Avec document, sans triche, sans copie sur les voisins, sans micro-ordinateur portable ou non, sans macro-ordinateur, sans téléphone portable ou non, sans oreillette de téléphone ni de dictaphone, sans talkie-walkie, sans télépathie, sans métempsycose, sans pompe. Sont tolérés : anti-sèche, tatouage ou vêtement imprimé en rapport avec le sujet, mouchoir de poche pré-imprimé, piercing ou scarification en rapport avec l'INF423, bronzage à code barre ou 2D...

```
    return &valeur;
}

int main(int argc, char **argv) {
    int valeur = 1;
    fonction(valeur);

    printf("%d\n", valeur);
    exit(0);
}
```

□

→

Question 2 : Que se passe-t-il lors de l'exécution du programme suivant ? (Durée ≈ 2 minutes)

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    char chaine[5];

    chaine[0] = 'H'; chaine[1] = 'e'; chaine[2] = 'l';
    chaine[3] = 'l'; chaine[4] = 'o';
    printf("%s\n", chaine);
}
```

□

→
→
→
→
→

Question 3 : Que se passe-t-il lors de l'exécution du programme suivant ? (Durée ≈ 2 minutes)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv) {
    char *chaine;

    chaine = strdup("Hello");
    chaine[2] = 0;
    printf("%s\n", chaine);
}
```

□

→
→
→
→
→

Question 4 : Qu'affiche le programme suivant ? (Durée ≈ 3 minutes)

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    printf("%d, %d, %d\n", 0x10, 011, 12);
    exit(0);
}
```

□

→

2 Stéganographie

Durée ≈ 25 minutes.

La stéganographie est la science (ou l'art) de dissimuler des secrets au sein de documents anodins, que ce soit des images, des fichiers musicaux, des documents textuels, ou même des programmes informatiques.

Dans cet exercice, on se propose de dissimuler un message au sein d'une image. Cette image sera représentée sous la forme d'une structure s_img définie ainsi :

```
struct s_img {
        int height, width;
        unsigned char *red;
        unsigned char *green;
        unsigned char *blue;
};
```

height et width contiennent respectivement la hauteur et la largeur de l'image et red, green et blue sont des tableaux contenant les valeurs des composantes rouges, vertes et bleues des différents pixels de l'image, les lignes de cette dernière étant disposées les unes à la suite des autres dans ces tableaux (la composante rouge du pixel situé à la position (x,y) est stockée à l'adresse red + y * width + x).

Le principe est de stocker un message en utilisant les deux bits de poids faible de chaque composante de chaque pixel de l'image. En effet, les variations introduites seront très faibles et donc peu perceptibles par l'oeil humain. Afin de simplifier l'écriture du programme, on ne modifiera que deux des trois composantes de couleur.

Vous disposez des deux fonctions suivantes qui permettent de lire et d'écrire une image sur le disque.

```
1 struct␣s_img␣*read_img(void);
2 void␣write_img(struct␣s_img␣*image);
```

Question 5 : Écrivez les fonctions :

```
1 void␣hide_message(unsigned␣char␣*message,␣struct␣s_img␣*image);
2 char*␣get_message(struct␣s_img␣*image);
```

qui permettent de cacher et de récupérer un message depuis une image ainsi qu'un petit programme simple les utilisant. □

3 Chaînes de caractères

Durée $\approx$ 20 minutes.

Nous allons implémenter une variante de la fonction strchr dont le but est de renvoyer les occurrences d'un caractère au sein d'une chaîne de caractères.

Voici la spécification de la fonction que vous devez coder :

```
1 char␣*␣find_char(char␣*chaine,␣char␣caractere);
```

Et voici comment elle doit se comporter :

La fonction renvoie un pointeur pointant vers la première occurrence du caractère caractère dans la chaîne chaine, ou NULL si le caractère n'est pas trouvé. En cas d'un appel ultérieur avec les mêmes paramètres (chaîne et caractère), la fonction renvoie la position de l'occurrence suivante du caractère dans la chaîne. En cas d'appel avec des paramètres différents, la fonction reprend son comportement normal (renvoie la position de la première occurrence du nouveau caractère dans la nouvelle chaîne passée en argument).

Question 6 : Écrivez cette fonction find_char. □

→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→

→
→
→