

Nom :	Année scolaire :	2007–2008
Prénom :	Date :	14 janvier 2008

Module INF₄₄₆
Session de juin

Programmation avancée en C

Contrôle de connaissance¹ 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'INF₄₄₆ !

1 Généralités

Question 1 : Quelle valeur le programme suivant affiche-t-il lors de l'exécution ?

```
1 #include <stdio.h>
   #include <stdlib.h>
3
   int i = 42;
5
   int main (int argc, char ** argv) {
```

¹Avec 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'INF₄₄₆, bronzage à code barre ou 2D...

```

7  printf ("%d\n", i);
   i++;
9  printf ("%d\n", i);
   {
11     int i = 8;
        printf ("%d\n", i);
13
        for (int i = 0; i < 2; i++)
15             printf ("%d\n", i);
           {
17             extern int i;
                printf ("%d\n", i);
19             }
                printf ("%d\n", i);
21     }
        printf ("%d\n", i);
23
        exit (EXIT_SUCCESS);
25 }

```

(3 minutes) □

→
→
→
→
→
→
→
→
→
→

2 Arithmétique binaire

Question 2 : Écrire une fonction qui calcule et renvoie la distance de HAMMING de 2 entiers passés en paramètre. Merci de vous adapter à la taille des `int` de l'architecture sur laquelle vous allez compiler votre programme. Si vous avez oublié, la distance de HAMMING entre 2 nombres est le nombre de bits qui diffèrent dans leur notation binaire. Par exemple la distance entre 5 (101_2) et 14 (1110_2) est 3. Cela sert à plein de choses (codage, moteurs de recherche, bio-informatique...). Pour vous inspirer, avez-vous remarqué que le « ou exclusif » en C $5 \wedge 14$ vaut 1011_2 ?

(10 minutes) □

→
→
→

3 Allocation

Question 3 : Qu'est-ce que le programmeur a voulu faire ? Quel est le problème concernant l'allocation² ?

```

1  #include <stdio.h>
   #include <stdlib.h>
3
   #define HEX(n, car) \
5  if ((n) > 9) \
   car = (n) - 10 + 'a'; \
7  else \
   car = (n) + '0';
9
   char *_code(unsigned_char_c){
11  char_t [3];
   HEX(c >> 4, t[0])
13  HEX(c & 15, t[1])
   t[2] = '\0';
15  return t;
   }
17
   int main(int argc, char** argv){

```

²Il y a plein d'autres problèmes, outre le fait que le programmeur aurait dû commenter son programme. ☺

```

19  _____unsigned_char_valeur_=_93;
    _____printf ("%s\n",_code ( valeur ));
21  _____exit (EXIT_SUCCESS);
    }

```

Le programme compile mais ne marche pas. Est-ce une coïncidence si le compilateur indique :

```

cc -std=c99    hexa.c    -o hexa
hexa.c: In function 'code':
hexa.c:15: attention : cette fonction retourne l'adresse
    d'une variable locale

```

(5 minutes) ☐

→
→
→
→
→
→
→
→
→
→

Question 4 : Que faire pour faire marcher l'exemple précédent ?

(3 minutes) ☐

→
→
→
→
→

4 Préprocesseur

Question 5 : Écrire dans le but d'aider la mise au point des programmes une macro-fonction `DEBUG_INT (n)` qui affiche la valeur d'une expression entière passée en paramètre et affiche le nom de la fonction, le nom de fichier et le numéro de ligne où se situe cette macro-fonction. Afin de supprimer les messages de débogage pour la version de production du programme, faire en sorte que si la macro-constante `NDEBUG` est définie, `DEBUG_INT (n)` soit vide³ (5 minutes) ☐

→
→

³C'est à dire que, si on compile avec l'option `-D NDEBUG` ou qu'on a un `#define NDEBUG` en tête de fichier, on veut supprimer les instructions de mise au point.

→
→
→
→
→
→
→
→

5 Petit problème sur les chaînes de caractères

Suite à des systèmes d'exploitation cassés, les machines du RésÉl n'ont plus d'éditeur de texte `vi` qui fonctionne et du coup les administrateurs systèmes sont tout perdus ! En particulier ils n'ont plus leur commande magique qu'ils connaissaient, la touche `s` qui permet de transformer une chaîne en une autre dans un texte. Ainsi, lorsqu'on tape dans `vi`

```
:%s/éditeur/Emacs/g
```

cela a pour effet de transformer toutes les occurrences du mot « éditeur » par le mot « Emacs » dans tout le texte.

Pour sauver tes camarades du RésÉl, tu vas devoir réaliser en C cette fonction sans laquelle ils ne savent plus rien faire !

À terme il faut réaliser la fonction

```
1 char *_substitue(char *_texte, char *_commande)
```

qui prend une chaîne de caractère `texte` sur laquelle on va travailler tel que

```
Je veux un éditeur !
```

```
Je vais l'appeler vi. vivivi mon petit éditeur adoré !
```

et une chaîne de caractères `commande` telle que `":%s/éditeur/poney/g"` et qui va donc renvoyer une nouvelle chaîne de caractères contenant :

```
Je veux un poney !
```

```
Je vais l'appeler vi. vivivi mon petit poney adoré !
```

Question 6 : Écris déjà une procédure

```
1 bool analyse_commande(char *_commande,
```

```
2 char **_avant,
  char **_après)
```

qui analyse la chaîne `commande` et renvoie dans le pointeur de caractère `avant` passé par adresse la première chaîne de la commande et dans `après` la seconde chaîne. Par exemple

Question 7 : Une fois qu'on a extrait avec la fonction précédente les chaînes à transformer, écrire la fonction

```
1 char*_substitue_chaine(char*_texte, _char*_avant, _char*_apres)
```

qui renvoie une chaîne contenant le `texte` où on a transformé toutes les occurrences des chaînes avant en après. (10 minutes) □

[illegible]

[illegible]

Durée totale estimée : 46 minutes.