

# Partiel de INF104, 23/02/2015

1h30 – Sans documents

Mise en garde : les questions de ce partiel supposent une lecture très attentive des énoncés, ainsi que des extraits de code fournis.

## **Questions sur le langage C (10 points)**

**Question 1 (1 point)** : On suppose qu'on a créé un fichier exécutable appelé `display_cl` à partir du code C fournit ci-dessous.

```
#include <stdio.h>

int main(int argc, char* argv[]){
    printf("value of argc: %d\n", argc);
    int i;
    for(i=0;i<argc;i++)
    {
        printf("value of argv[%d]: %s\n", i, argv[i]);
    }
    return 0;
}
```

On exécute la commande : `./display_cl -c -d 5`

Quel affichage obtient-on ? Justifier votre réponse en précisant les valeurs prises par les paramètres `argc` et `argv`.

**Question 2 (2 points)** : En considérant un programme dont le code est fourni ci-dessous, qu'est-ce qui va s'afficher à l'écran lors de son exécution ? Justifiez votre réponse. Pour information, la représentation en hexadécimal de 257 est : `0x0101`. Nous rappelons ici qu'en hexadécimal, les chiffres sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, et F.

```
#include <stdio.h>

void abs(int x)
{
    if(x<0)
        x=-x;
}

int main(int argc, char* argv[]){
    int x = 5;
    abs(x);
    printf("value of x: %d\n", x);
    x=-5;
    abs(x);
    printf("value of x: %d\n", x);
    char c = 257;
    printf("value of c: %d\n", c);
    return 0;
}
```

**Question 3 (1.5 point)** : Qu'est-ce qu'une fuite mémoire ? Donner la signature de la fonction C qui permet de réaliser une allocation dynamique de mémoire en C.

**Question 4 (1.5 point) :** On s'intéresse au code C suivant :

```
#include <stdio.h>

short glob_var[5] = {257,255,360,365,460};

int main(int argc, char* argv[]){
    char * var1 = (char*) glob_var;
    short * var2 = (short*) var1;

    printf("size of short: %d\n", (char) sizeof(short));
    printf("value of var1, hexa: %p\n",var1);
    printf("value of var2, hexa: %p\n", (char*) var2);

    var1++;

    printf("value of var1, hexa: %p\n", (char*) var1);

    var2++;

    printf("value of var2, hexa: %p\n", (char*) var2);
    printf("value of *var2, hexa: %04x\n", *var2);
    printf("value of *var1, hexa: %04x\n", *var1);
    return 0;
}
```

*Indication sur le code :* %p permet d'afficher une adresse mémoire sous la forme d'un nombre hexadécimal. Nous rappelons ici qu'en hexadécimal, les chiffres sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, et F. %04x permet d'afficher une valeur avec 4 chiffres en base hexadécimal. Par exemple, l'affichage de la valeur 8 avec le format %04x est : 0008.

On vous donne une partie de l'affichage obtenue lors de l'exécution du programme ci-dessus, à vous de le compléter (en remplaçant les ...) par la valeur qui devrait s'afficher.

```
size of short: 2
value of var1, hexa: C528
value of var2, hexa: C528
value of var1, hexa: C529
value of var2, hexa: ...
value of *var2, hexa: ...
value of *var1, hexa: ...
```

**Question 5 (4 points) :** Ecrire le code C qui implémente la fonction **strcpy**. Pour information, la fonction **strcpy** copie la chaîne de caractères `src` dans la chaîne de caractère `dst` (y compris le caractère de fin de chaîne : `'\0'`).

Il faut donc que vous complétiez le code ci dessous (en remplaçant les ...) :

```
void strcpy(char *dst, const char *src)
{
    ...
}
```

## **Questions sur la chaîne de production (10 points)**

**Question 6 (2 points):** Expliquez brièvement la différence entre la compilation et l'édition de lien. Quelle option de **gcc** permet de s'arrêter avant l'édition de lien ?

Considérons le programme C suivant :

Contenu du fichier main.c
<pre>#define max 100; #define double(x) 2*(x) #include &lt;stdio.h&gt;  int sup( int n, int m) ;  int main (int argc, char* argv[]){     printf("%d", sup (double(argc), max) );     return 0 ; }  int sup(int n1, int n2) {     if(n1&gt;n2)         return n1;     return n2; }</pre>

**Question 7 (1 point):** Ecrivez le code résultant de l'exécution du pré-processeur sur le fichier main.c ci-dessus. Attention, on ignorera la traduction du `#include <stdio.h>`.

**Question 8 (1 point):** Après exécution du pré-processeur, la compilation va-t-elle réussir ou échouer dans le cas du code ci-dessus? Justifiez votre réponse.

On souhaite compiler le programme constitué des fichiers sources suivants :

Contenu du fichier main.c	Contenu du fichier globs.c
<pre>char state=0; void print_state();  int main(int argc, char* argv[]){     print_state();     return 0; }</pre>	<pre>#include &lt;stdio.h&gt;  char state=0; void print_state() {     printf("State=%d",state); }</pre>

**Question 9 (1.5 point):** On va exécuter la commande :

```
gcc -o appli main.c globs.c
```

Pendant, on sait qu'on va avoir une erreur... En quoi consiste cette erreur? A quelle étape de l'exécution de **gcc** va-t-elle se manifester ?

**Question 10 (1.5 point):** Comment modifier le code précédent pour résoudre ce problème ? Indiquez, pour un fichier donné, la (ou les) ligne(s) de code que vous devez supprimer et le code que vous utilisez en remplacement des lignes de code enlevées.

**Question 11 (2 points):** on souhaite écrire un makefile qui compile un programme C à partir de trois fichiers C : main.c, graph.c, path\_computation.c. Compléter le makefile suivant en remplaçant les pointillés par le code qui convient.

```
#####
CC      = gcc
CFLAGS  = ...
LD      = gcc
OFILES  = main.o graph.o path_computation.o
#####

clean:
    rm *.o prog

all: prog

#production de l'objet main.o (a completer)
main.o: ...
    $(CC) $(CFLAGS) ...

# production de l'objet graph.o (a completer)
graph.o: ...
    $(CC) $(CFLAGS) ...

# production de l'objet path_computation.o (a completer)
path_computation.o: ...
    $(CC) $(CFLAGS) ...

# production de l'executable prog (a completer)
prog: $(OFILES)
    $(LD) ... -o prog
```

**Question 12 (1 point):** Avec le Makefile ci dessus, quel sera l'effet de l'exécution de la commande : make ?