M1 MOSIG
Programing with C

Advanced

2015-2016

In the following, you are given example programs and some simple problems. The goal is to understand, program (if needed), compile and run the programs. Do not hesitate to ask questions and have fun !

Exercice 1 : Parameter Passing

What do you think of this program ? Is it correct ?

```
void exchange (int  a ,int b)
{
  int t = a;
  a=b;
  b=t;
}
int main(int argc, char** argv) {
  int x=10, y=20;
  exchange(x,y);
  printf("x=%d, y=%d\n", x, y);
}
```

Exercice 2 : Arrays and Pointers

Complete the two following code snippets.

```
...
int tab[] = {1, 2, 3, 4, 5, -1};
for (int *p = tab; .../* complete */ ...) {
  printf("Address: %p, value : %i\n", p, *p);
}

int tab2[5] = {1, 2, 3, 4, 5};
int *p2 = tab2;
for (int n = 0; n < 5; n++) {
  printf("Address %p, value %i\n", p2, *p2);
  /* ... complete ...*/
}
```

Exercice 3 : String Manipulation

▷ **Question 1.** *Write a program that counts the number of letters in a string. Write two versions : one, using the standard library (`#include <string.h>`), and another without it.*

Exercice 4 : What happens ?

```
char *str = malloc(7);
strcpy(str,"azerty");
printf("%s\n", str);
free(str);
printf("%s\n", str);
char *c = malloc(7);
printf("%c\n", str);
```

▷ **Question 1.** *You can try valgrind on this example with `valgrind <your executable>`. Your program should be compiled with the `-g` option.*

Exercice 5 : Copy of memory zones

```
const int size = 5;
int *tab = calloc(size , sizeof(int));
int *copy = malloc( size*sizeof(int ));
memcpy(copy , tab , size*sizeof(int ));
for (int i = 0; i < size; i++)
  printf("%i ", copy[i]);
printf(" \n ") ;
```

▷ **Question 1.** *What does the memcpy function ?*

▷ **Question 2.** *What is the difference between `malloc` and `calloc` ?*

Exercice 6 :

```
#include <assert.h>
...
while (1) {
  int *boom = malloc(1 << 30);
  assert(boom != NULL);
}
```

Exercice 7 : Implementing a chained list

```
struct cell {
  int val ;
  struct cell *next ;
};

//print the list
void print ( struct cell *l ) {
  while (l != NULL) {
```

```
    printf("%i ->", l->val);
    l = l->next;
  }
  printf( "END\n " ) ;
}

//insert
void insert_front (struct cell **pl , int v) {
  struct cell *cl = malloc(sizeof(struct cell));
  assert(c != NULL);
  c->val = v;
  c->next = *pl;
  *pl = c;
}
```

▷ **Question 1.** *Why is there a double star (`struct cell **pl`) for the `insert_front` function ?*

▷ **Question 2.** *Write the `insert_end` functon which inserts an element at the end of the list*

▷ **Question 3.** *Write the `delete_front` function which deletes the first element of the list.*

▷ **Question 4.** *Write the `delete_val` function which deletes the first cell containing a given value.*

▷ **Question 5.** *You may test with*

```
int main(void) {
  struct cell *list = NULL; // important!
  for (int i = 6; i < 10; i++) {
    insert_end(&list , i );
    print(list);
  }
  for (int i = 5; i > 0; i--) {
    insert_front(&list, i);
    print(list);
  }
  srand ( time (NULL) ) ;
  while (list != NULL) {
    delete_front(&list);
    if (list != NULL) delete_val(&list, rand()%10);
    print(list);
  }
  return 0;
}
```

Exercice 8 : What is the bug ?

```
#include <stdio.h>

int main() {
  int *age;

  printf("Hello\n");
  printf("What is your age\n");
```

```
  scanf("%d", age);
  printf("Your are %d years old\n", *age);
  return 0;
}
```

Exercice 9 : A problem ?

```
#include <stdio.h>
int main()
{
  unsigned int i;
  for (i = 12; i >= 0; --i) {
    printf("i = %d\n", i);
  }
}
```

Exercice 10 : Pointers and Arrays

```
#include <stdio.h>

int *a, *b;

void f(int x) {
  int i[3];
  i[0] = x;
  i[1] = x + 1;
  i[2] = x + 2;
  a = i;
}

void g(int x) {
  int i[3];
  i[0] = x;
  i[1] = x + 1;
  i[2] = x + 2;
  b = i;
}

int main() {
  f(1); /* Modify a */
  printf("a = {%d,%d,%d}\n", a[0], a[1], a[2]);
  g(2); /* Modify b */
  printf("a = {%d,%d,%d}\n", a[0], a[1], a[2]);
  return 0;
}
```

▷ **Question 1.** *Change the program so as to have only one function for modifying an array (ant not two almost identical ones, `f` and `g`).*

Exercice 11 : On Function Pointers

If your compiler has all the warnings turned on, it will certainly tell you if there is a problem...

```
#include <stdio.h>

const int TRUE = 1;
const int FALSE = 0;

int function_returning_false() {
  return FALSE;
}

int main() {
  if (function_returning_false) {
    printf("function returned true\n");
  }
}
```

Exercice 12 : What about this for loop ?

```
#include <stdio.h>
int main() {
  int i;
  for (i = 0; i <= 42; i++);
  {
    printf("i=%d\n", i);
  }
  return 0;
}
```

Exercice 13 : C is not Java

```
#include <stdio.h>

int main()
{
  char *hello = "hello, world!" + 3;
  char *charstring = 'h' + "ello, world!";

  printf("hello=%s, charstring=%s.\n", hello, charstring);
  printf("Eh non, on n'est pas en Java !\n");
  return 0;
}
```

Exercice 14 : Linked Lists

▷ **Question 1.** *Define the data structure and the functions for a double linked list i.e a list in which each element points both to the previous and the next element.*