Assignment 9

Exercise 00: Makefile

Turn-in files	Makefile
Allowed functions	nothing

- Create the **Makefile** that will compile your **libvc.a** from assignment 6.
- The **Makefile** will get its source files from the "srcs" directory.
- The **Makefile** will get its header files from the "includes" directory.
- The **Makefile** should also implement the following rules: **clean**, **fclean** and **re** as well as **all**.
- **fclean** does the equivalent of a make clean and also erases the binary created during the make. **re** does the equivalent of a make **fclean** followed by a make.

Exercise 01: vc_foreach

Turn-in files	vc_foreach.c
Allowed functions	nothing

- Create the function **vc_foreach** which, for a given ints array, applies a function on all elements of the array. This function will be applied following the array's order.
- Here's how the function should be prototyped :

```
void vc_foreach(int *tab, int length, void(*f)(int));
```

- For example, the function **vc_foreach** could be called as follows in order to display all 86 ints of the array :

```
vc_foreach(tab, 86, &vc_putnbr);
```

Exercise 02:vc_map

Turn-in files	vc_map.c
Allowed functions	malloc

- Create the function **vc_map** which, for a given ints array, applies a function on all elements of the array (in order) and returns a array of all the return values. This function will be applied following the array's order.
- Here's how the function should be prototyped :

```
int *vc_map(int *tab, int length, int(*f)(int));
```

Exercise 03: vc_any

Turn-in files	vc_any.c
Allowed functions	nothing

- Create a function **vc_any** which will return 1 if, passed to the function f, at least one element of the array returns 1. Else, it should return 0.
- Here's how the function should be prototyped :

```
int vc_any(char **tab, int(*f)(char*));
```

- The array will be delimited by 0.

Exercise 04: vc_count_if

Turn-in files	vc_count_if.c
Allowed functions	nothing

- Create a function **vc_count_if** which will return the number of elements of the array that return 1, passed to the function f.
- Here's how the function should be prototyped :

```
int vc_count_if(char **tab, int(*f)(char*));
```

- The array will be delimited by 0.

Exercise 05: vc_is_sort

Turn-in files	vc_is_sort.c
Allowed functions	nothing

- Create a function **vc_is_sort** which returns 1 if the array is sorted and 0 if it isn't.
- The function given as argument should return a negative integer if the first argument is lower than the second, 0 if they're equal or a positive integer for anything else.
- Here's how the function should be prototyped :

```
int vc_is_sort(int *tab, int length, int(*f)(int, int));
```

Exercise 06: calc

Turn-in files	Makefile, calc.c

Turn-in files	Makefile, calc.c
Allowed functions	vc_putnbr, vc_putstr, vc_atoi

- Create a program called **calc**.
- The progam will be executed with three arguments: calc value1 operator value2
- Example :

```
$ ./calc 11 "+" 21
32
```

- The operator character corresponds to the appropriate function within an array of pointers to function.
- Your directory should contain a Makefile with the all and clean rules.
- In the case of an **invalid argument** such as ./calc foo divide bar, the program returns **0**.
- If the number of arguments is invalid, calc doesn’t display anything.
- Here’s an example of tests that will be run:

```
$ make clean
$ make
$ ./calc
$ ./calc 1 + 1
2
$ ./calc 50amis - -20toto12
70
$ ./calc 1 p 1
0
$ ./calc 1 + toto3
1
$
$ ./calc toto3 + 4
4
$ ./calc foo plus bar
0
$ ./calc 25 / 0
Stop : division by zero
$ ./calc 3 / 1
3
$ ./calc 8 * 6
48
$ ./calc 25 % 0
Stop : modulo by zero
```

Exercise 07: vc_sort_words

Turn-in files	vc_sort_words.c
Allowed functions	vc_strcmp

- Create the function vc_sort_words, which sorts words obtained with **vc_split_whitespaces** by **ascii** order.
- The sorting will be performed by exchanging the array's pointers.
- Here's how it should be prototyped :

```
void vc_sort_words(char **words);
```