Exercise 6

1. Pointers and Arrays + Stack and Heap Segments

Draw the stack and heap segments just when the PC register points to the last semicolon ; of the following compound statements (assuming that local arrays are stored in the heap segment):

1.

```c
int i;
int j = 1;
int *p = &j
int**q = &p
int ***r = &q
i = ***r + 1;
```

2.

```c
int i;
int tab[3];
int *p = tab;
++p;
++p;
i = p - tab;
tab[0] = 1;
(tab+1)[0] = 2;
*p = 3;
```

3.

```c
int *p = malloc(2*sizeof(int));
p[0]=4;
p[2]=5;
```

2. Complicated Declarations

Translate the following declarations into a natural language description in **your mother tongue**.

Example (for English):

`char **argv`: argv is a pointer to a pointer to a char

`char *x()`: x is a function returning a pointer to a char

1. `int a[3][2];`
2. `int *b[3];`
3. `int (*c)();`
4. `int d(int (*e)());`
5. `int (*f())();`

Hint: To translate into English, you can use the following website: https://cdecl.org

3. typedef Definitions

The following `typedef` define some very common new types. Indicate their names and their corresponding defined types.

Example:

typedef int my__type: my__type is a synonym to int.

1. `typedef void *stackt;`
2. `typedef int (*fctInt\_t)(int);`

3. `typedef void *(*fct\_gen)(void *);`
4. `typedef void ( *signal(int, void (*)(int)))(int);`

Hint: The last one is a bit tricky. Decompose this complicated definition in two more readable ones.

4. Pointer to Function + typedef

Consider the following program:

```c
typedef int (*mathFunc\_t)(int, int); // definition of type mathFunc\_t

int add(int a, int b) {
 return a + b;
}

int mult(int a, int b) {
 return a * b;
}

int compute(mathFunc\_t f, int a, int b) {
 return f(a,b);
}

int main() {
 mult(add(2,4), 8);
 compute(mult, compute(add,2,4), 8);
 return 0;
}
```

1. What is the return value of `mult()` and `compute()`?
2. Draw the simplified stack segments when the PC register points respectively to the last semicolon ; of the function calls in line 1 and 2 of `main()`.

Hint: There are 2 fct calls in the first line of `main()`, and 4 in the second; if you are lost, consult the solution of ex. 2, Series 5.

5. Project P01: Linked Data In-Memory Store

2. Study the project description.
3. Build the groups and create one common repository for the project on GitLab.
4. To initialize your repository, make a first commit with a README.md file.