C programming excercices 1

1 Beginner excercises

1. Create a hello world example. Then build it with CMake.
2. Create a hello world example with defineing two macros. One beeing `HELLO`, the other `WORLD`
3. Lest check the default value of a variable. Create a variable and print out its value, without giving it one.
4. Print out the address and the value of the previous variable in the following format:

```
[address]:    value
[0xABABABAB]: 00123
```

5. Print out the address and the value of a floating-point variable in the following format:

```
[0xABABABAB]: 012.345
```

2 Header and source file excercises

Place all of the following excercises in separate sections and functions. With using other source and header file. *The excercises can go into the same source and header files.*

1. Create a variable containing the following string: `"The sky is beautiful!"`. Print it out until the 10th character with a function.
2. Replace the `'e'` characters in the previous string to the `'X'` character. Then print it.
3. Create a struct with the following parameters `"int index; int value;"`. Define it as a type. (Use `typedef existing_type new_type;`)
   Create an array of the previously defined type with 10 elements, and fill it up with the square of 1 to 10. Print out the values of the array after filling it up.
4. Create the function `void square(int *a)` and square the incoming pointers value. Print out the value of A before and after executing said function.

3 Harder excercises

Put every excercise in their separate .c and .h file. Include the headers from the main program and compile with CMake.

1. Create character array, with 100 elements. Ask the user for a text input, and read it into the created array. Count the characters given by the user and copy the content into a new char "array", which has the exact size of the input text. *Make sure to also free the new array before exiting the program.*

2. Create a function `void replacer(char* str)`, which will write `"FILTERED"` over the second word of the text. Make sure to don't overflow the original variable. The text can cover multiple words, until the end of the incoming string, but only need to be written once. Print out the incoming string before and after the function execution.

3. Iterate trough the following array with a pointer until we are at the last elemnt, that is containing a `NULL (0)` value:

```c
unsigned int myarray[] = { 1, 2, 3, 4, 5, 11, 22, 33, 44, 55, 0 };
```

   and print out its values.

4. Create a simple Linked List. Complete the following code and test your approach.

```c
struct list {
    int item;
    struct list* next;
};

struct list* list_create() {
    /* TODO */
}
int list_add(struct list*, int item) {
    /* TODO */
}
int list_remove(struct list*, int index) {
    /* TODO */
}
int list_get(struct list*, int index) {
    /* TODO */
}
struct list* list_get_next(struct list*) {
    /* TODO */
}

int main() {
    struct list* my_list = list_create();
    /* TODO */
}
```