



B1- Unix and C Lab Seminar

B-CPE-100

Day 08

Compilation, Allocation

v2.0



Day 08

Compilation, Allocation

repository name: CPool_Day08_\$ACADEMICYEAR

repository rights: ramassage-tek

language: C

group size: 1



- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- Don't push your **main** function into your delivery directory, we will be adding our own. Your files will be compiled adding our **main.c**.
- If one of your files prevents you from compiling with *.c, the Autograder will not be able to correct your work and you will receive a 0.



All .c files from your delivery folder will be collected and compiled with your **libmy**, which is found in **CPool_Day08_\$ACADEMICYEAR/lib/my**. For those of you using .h files, they must be located in **CPool_Day08_\$ACADEMICYEAR/include**.

Some tests will automatically compile your functions the following way:

```
Terminal
~/B-CPE-100> cd task01
~/B-CPE-100> cc *.c -c -I../include/
~/B-CPE-100> cc *.o ~autograder/main_task01.o -L../lib/my/ -o task01 -lmy
```



Create your repository at the beginning of the day and submit your work on a regular basis!
The delivery directory is specified within the instructions for each task.
In order to keep your repository clean, pay attention to **gitignore**.



Allowed system function(s): write, malloc, free



We still encourage you to write unit tests for all your functions!
Check out Day06 if you need an example, and re-read this document.



Task 01

my_strdup

Write a function that allocates memory and copies the string given as argument in it. It must be prototyped as follows:

```
char *my_strdup(char const *src);
```

The function must return a pointer to the newly allocated string.

Delivery: CPool_Day08_\$ACADEMICYEAR/my_strdup.c

Task 02

concat_params

Write a function that turns the command-line given arguments into a single string. Arguments are to be separated by '\n'. The function will be called the following way:

```
int main(int ac, char **av)
{
    my_putstr(concat_params(ac, av));
    return (EXIT_SUCCESS);
}
```

```
Terminal
~/B-CPE-100> cc -o concat_params concat_params.c main.c -L./lib/my -lmy
~/B-CPE-100> ./concat_params toto titi | cat -e
./concat_params$
toto$
titi
```

The function must be prototyped as follows:

```
char *concat_params(int argc, char **argv);
```

Delivery: CPool_Day08_\$ACADEMICYEAR/concat_params.c



Task 03

my_show_word_array

Write a function that displays the content of an array of words.
There must be one word per line, and each word must end with '\n', including the last one.
The function must be prototyped as follows:

```
int my_show_word_array(char * const *tab);
```

Delivery: CPool_Day08_\$ACADEMICYEAR/my_show_word_array.c

Here is an example of main function:

```
int main()
{
    char *test_word_array[] = {"The", "Answer", "to", "the", "Great", "Question ...", "Of",
                               "Life", "the", "Universe", "and", "Everything ...", "Is ...", "Forty-two", 0};

    my_show_word_array(test_word_array);
}
```

Task 04

my_str_to_word_array

Write a function that splits a string into words. Separators will all be non-alphanumeric characters.
The function returns an array in which each cell contains the address of a string (representing a word).
The last cell must be null to terminate the array.
The function must be prototyped as follows:

```
char **my_str_to_word_array(char const *str);
```

Delivery: CPool_Day08_\$ACADEMICYEAR/my_str_to_word_array.c



Task 05

convert_base

Write a function that returns the result from the **nbr** string conversion (expressed in a **base_from** radix to a **base_to** radix), in the form of a newly, and sufficiently, allocated string.

The number, represented by **nbr**, fits in an integer.

The function must be prototyped as follows:

```
char *convert_base(char const *nbr, char const *base_from, char const *base_to);
```

Delivery: CPool_Day08_\$ACADEMICYEAR/convert_base.c