B1- Unix and C Lab Seminar

B-CPE-100

Day 06

Pointers are back

v1.2

Day 06

Pointers are back

repository name:	CPool_Day06_$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** and our **my_putchar.c** files.
- You are only allowed to use the **my_putchar** function to complete the following tasks, but don't push it into your delivery directory, and don't copy it in *any* of your delivered files.
- 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.

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`.

Most of the day's functions exist in the **string** library. Use **man** to obtain a full explanation of how a function works. Beware that none of your deliveries contains a function from this **string** library!

Don't forget to write unit tests for all your functions!
Check out Day04 if you need an example, and re-read this document.

Task 01

my_strcpy

Write a function that copies a string into another. The destination string will already have enough memory to copy the source string.
It must be prototyped the following way:

```
char *my_strcpy(char *dest, char const *src)
```

The function returns `dest`.
Delivery: CPool_Day06_$ACADEMICYEAR/my_strcpy.c

Task 02

my_strncpy

Write a function that copies n characters from a string into another.
The destination string will already have enough memory to contain n characters.
It must be prototyped the following way:

```
char *my_strncpy(char *dest, char const *src, int n);
```

The function returns `dest`.
Delivery: CPool_Day06_$ACADEMICYEAR/my_strncpy.c

Add '\0's if n is strictly greater than the length of the string. Do not add '\0' if n is strictly lower than the length of the string (because `dest` is not supposed to contain more than n bytes.

Task 03

my_revstr

Write a function that reverses a string. It must be prototyped the following way:

```
char *my_revstr(char *str);
```

The function returns `str`.
Delivery: CPool_Day06_$ACADEMICYEAR/my_revstr.c

Task 04

my_strstr

Reproduce the behavior of the *strstr* function. Your function must be prototyped the following way:

```
char *my_strstr(char const *str, char const *to_find);
```

Delivery: CPool_Day06_$ACADEMICYEAR/my_strstr.c

Check out the *my_strcmp* and *my_strncmp* functions.

Task 05

my_strcmp

Reproduce the behavior of the *strcmp* function. Your function must be prototyped the following way:

```
int my_strcmp(char const *s1, char const *s2);
```

Delivery: CPool_Day06_$ACADEMICYEAR/my_strcmp.c

Task 06

my_strncmp

Reproduce the behavior of the *strncmp* function. Your function must be prototyped the following way:

```
int my_strncmp(char const *s1, char const *s2, int n);
```

The function should return the same values as *strcmp(3)*.
Delivery: CPool_Day06_$ACADEMICYEAR/my_strncmp.c

Task 07

my_strupcase

Write a function that puts every letter of every word in it in uppercase.
It must be prototyped the following way:

```
char *my_strupcase(char *str);
```

The function returns *str*.
Delivery: CPool_Day06_$ACADEMICYEAR/my_strupcase.c

Task 08

my_strlowcase

Write a function that puts every letter of every word in it in lowercase.
It must be prototyped the following way:

```
char *my_strlowcase(char *str);
```

The function returns *str*.
Delivery: CPool_Day06_$ACADEMICYEAR/my_strlowcase.c

Task 09

my_strcapitalize

Write a function that capitalizes the first letter of each word.
It must be prototyped the following way:

```
char *my_strcapitalize(char *str);
```

The function returns *str*.
Delivery: CPool_Day06_$ACADEMICYEAR/my_strcapitalize.c

The sentence, *hey, how are you? 42WORds forty-two; fifty+one* will become *Hey, How Are You? 42words Forty-Two; Fifty+One*.

Task 10

my_str_isalpha

Write a function that returns **1** if the string passed as parameter only contains alphabetical characters and **0** if the string contains another type of character. It must be prototyped the following way:

```
int my_str_isalpha(char const *str);
```

The function returns **1** if *str* is an empty string.
Delivery: CPool_Day06_$ACADEMICYEAR/my_str_isalpha.c

Task 11

my_str_isnum

Write a function that returns **1** if the string passed as parameter only contains digits and **0** otherwise.
It must be prototyped the following way:

```
int my_str_isnum(char const *str);
```

The function returns **1** if *str* is an empty string.
Delivery: CPool_Day06_$ACADEMICYEAR/my_str_isnum.c

Task 12

my_str_islower

Write a function that returns **1** if the string passed as parameter only contains lowercase alphabetical characters and **0** otherwise.
It must be prototyped the following way:

```
int my_str_islower(char const *str);
```

The function returns **1** if the *str* is an empty string.
Delivery: CPool_Day06_$ACADEMICYEAR/my_str_islower.c

Task 13

my_str_isupper

Write a function that returns **1** if the string passed as parameter only contains uppercase alphabetical characters and **0** otherwise.
It must be prototyped the following way:

```
int my_str_isupper(char const *str);
```

The function returns **1** if *str* is an empty string.
Delivery: CPool_Day06_$ACADEMICYEAR/my_str_isupper.c

Task 14

my_str_isprintable

Write a function that returns **1** if the string passed as parameter only contains printable characters and **0** otherwise.
It must be prototyped the following way:

```
int my_str_isprintable(char const *str);
```

The function returns **1** if *str* is and empty string.
Delivery: CPool_Day06_$ACADEMICYEAR/my_str_isprintable.c

man isprint

Task 15

my_putnbr_base

Write a function that converts and displays a decimal number into a number in a given base.
The number is given as an *int* and the base is provided as a *string*.
The base contains all the symbols that can be used to print a number (for instance, *0123456789* for the decimal base, *01* for the binary base, *0123456789ABCDEF* for the hexadecimal base).
The function must deal with negative numbers, and be prototyped the following way:

```
int my_putnbr_base(int nbr, char const *base);
```

Delivery: CPool_Day06_$ACADEMICYEAR/my_putnbr_base.c

Task 16

my_getnbr_base

Write a function that converts and returns a number (provided as a *string*) in a given base into a decimal number.
The function must deal with negative numbers, and several successive + or - before the number.
If any error occurs, the function must return **0**. It must be prototyped the following way:

```
int my_getnbr_base(char const *str, char const *base);
```

Delivery: CPool_Day06_$ACADEMICYEAR/my_getnbr_base.c

Task 17

my_showstr

Write a function that prints a string and returns **0**. If this string contains non-printable characters, they must be printed hexadecimally (in lowercase letters) with a backslash before the given value.
It must be prototyped the following way:

```
int my_showstr(char const *str);
```

Delivery: CPool_Day06_$ACADEMICYEAR/my_showstr.c

For instance, *I like \n ponies!* will be printed as *I like \0a ponies!*.

Task 18

my_showmem

Write a function that prints a memory dump and return **0**. It must be prototyped the following way:

```
int my_showmem(char const *str, int size);
```

Each line of the output manages 16 characters and is divided into three columns:

- The hexadecimal address of the line's first character,
- the content in hexadecimal,
- the content in printable characters.

Any non printable characters must be replaced by a dot.

Terminal

```
~/B-CPE-100> ./my_showmem | cat -e
00000000:  6865 7920 6775 7973 2073 686f 7720 6d65 hey guys show me$
00000010:  6d20 6973 2063 6f6f 6c20 796f 7520 6361 m is cool you ca$
00000020:  6e20 646f 2073 6f6d 6520 7072 6574 7479 n do some pretty$
00000030:  206e 6561 7420 7374 7566 6600 0f1b 7f05  neat stuff.....$
00000040:  2e00 0102 0304 0506 0708 090e 0f1b 7f   ..............$
```

Delivery: CPool_Day06_$ACADEMICYEAR/my_showmem.c

Don't forget the padding if there aren't enough characters to have a valid alignment.