V.3 Part 2 - Additional functions

In this second part, you must code a set of functions that are either not included in the `libc`, or included in a different form. Some of these functions can be useful to write Part 1's functions.

ft_memalloc	
Prototype	`void * ft_memalloc(size_t size);`
Description	Allocates (with `malloc(3)`) and returns a "fresh" memory area. The memory allocated is initialized to `0`. If the allocation fails, the function returns `NULL`.
Param. #1	The size of the memory that needs to be allocated.
Return value	The allocated memory area.
Libc functions	`malloc(3)`

ft_memdel	
Prototype	`void ft_memdel(void **ap);`
Description	Takes as a parameter the address of a memory area that needs to be freed with `free(3)`, then puts the pointer to `NULL`.
Param. #1	A pointer's address that needs its memory freed and set to `NULL`.
Return value	None.
Libc functions	`free(3)`.

ft_strnew	
Prototype	`char * ft_strnew(size_t size);`
Description	Allocates (with `malloc(3)`) and returns a "fresh" string ending with `'\0'`. Each character of the string is initialized at `'\0'`. If the allocation fails the function returns `NULL`.
Param. #1	The size of the string to be allocated.
Return value	The string allocated and initialized to `0`.
Libc functions	`malloc(3)`

ft_strdel	
Prototype	`void ft_strdel(char **as);`
Description	Takes as a parameter the address of a string that need to be freed with `free(3)`, then sets its pointer to `NULL`.
Param. #1	The string's address that needs to be freed and its pointer set to `NULL`.
Return value	None.
Libc functions	`Free(3)`.

ft_strclr	
Prototype	`void ft_strclr(char *s);`
Description	Sets every character of the string to the value `'\0'`.
Param. #1	The string that needs to be cleared.
Return value	None.
Libc functions	None.

ft_striter	
Prototype	`void ft_striter(char *s, void (*f)(char *));`
Description	Applies the function `f` to each character of the string passed as argument. Each character is passed by address to `f` to be modified if necessary.
Param. #1	The string to iterate.
Param. #2	The function to apply to each character of `s`.
Return value	None.
Libc functions	None.

ft_striteri	
Prototype	`void ft_striteri(char *s, void (*f)(unsigned int, char *));`
Description	Applies the function `f` to each character of the string passed as argument, and passing its index as first argument. Each character is passed by address to `f` to be modified if necessary.
Param. #1	The string to iterate.
Param. #2	The function to apply to each character of `s` and its index.
Return value	None.
Libc functions	None.

ft_strmap	
Prototype	`char * ft_strmap(char const *s, char (*f)(char));`
Description	Applies the function `f` to each character of the string given as argument to create a "fresh" new string (with `malloc(3)`) resulting from the successive applications of `f`.
Param. #1	The string to map.
Param. #2	The function to apply to each character of `s`.
Return value	The "fresh" string created from the successive applications of `f`.
Libc functions	`malloc(3)`

ft_strmapi	
Prototype	`char * ft_strmapi(char const *s, char (*f)(unsigned int, char));`
Description	Applies the function `f` to each character of the string passed as argument by giving its index as first argument to create a "fresh" new string (with `malloc(3)`) resulting from the successive applications of `f`.
Param. #1	The string to map.
Param. #2	The function to apply to each character of `s` and its index.
Return value	The "fresh" string created from the successive applications of `f`.
Libc functions	`malloc(3)`

ft_strequ	
Prototype	`int ft_strequ(char const *s1, char const *s2);`
Description	Lexicographical comparison between `s1` and `s2`. If the 2 strings are identical the function returns `1`, or `0` otherwise.
Param. #1	The first string to be compared.
Param. #2	The second string to be compared.
Return value	`1` or `0` according to if the 2 strings are identical or not.
Libc functions	None.

ft_strnequ	
Prototype	`int ft_strnequ(char const *s1, char const *s2, size_t n);`
Description	Lexicographical comparison between `s1` and `s2` up to `n` characters or until a `'\0'` is reached. If the 2 strings are identical, the function returns `1`, or `0` otherwise.
Param. #1	The first string to be compared.
Param. #2	The second string to be compared.
Param. #3	The maximum number of characters to be compared.
Return value	`1` or `0` according to if the 2 strings are identical or not.
Libc functions	None.

ft_strsub	
Prototype	`char * ft_strsub(char const *s, unsigned int start, size_t len);`
Description	Allocates (with `malloc(3)`) and returns a "fresh" substring from the string given as argument. The substring begins at index`start` and is of size `len`. If `start` and `len` aren't refering to a valid substring, the behavior is undefined. If the allocation fails, the function returns `NULL`.
Param. #1	The string from which create the substring.
Param. #2	The start index of the substring.
Param. #3	The size of the substring.
Return value	The substring.
Libc functions	`malloc(3)`

ft_strjoin	
Prototype	`char * ft_strjoin(char const *s1, char const *s2);`
Description	Allocates (with `malloc(3)`) and returns a "fresh" string ending with `'\0'`, result of the concatenation of `s1` and `s2`. If the allocation fails the function returns `NULL`.
Param. #1	The prefix string.
Param. #2	The suffix string.
Return value	The "fresh" string result of the concatenation of the 2 strings.
Libc functions	`malloc(3)`

ft_strtrim	
Prototype	`char * ft_strtrim(char const *s);`
Description	Allocates (with `malloc(3)`) and returns a copy of the string given as argument without whitespaces at the beginning or at the end of the string. Will be considered as whitespaces the following characters `' '`, `'\n'` and `'\t'`. If `s` has no whitespaces at the beginning or at the end, the function returns a copy of `s`. If the allocation fails the function returns `NULL`.
Param. #1	The string to be trimed.
Return value	The "fresh" trimmed string or a copy of `s`.
Libc functions	`malloc(3)`

ft_strsplit	
Prototype	`char ** ft_strsplit(char const *s, char c);`
Description	Allocates (with `malloc(3)`) and returns an array of "fresh" strings (all ending with `'\0'`, including the array itself) obtained by spliting `s` using the character `c` as a delimiter. If the allocation fails the function returns `NULL`. Example : `ft_strsplit("*hello*fellow***students*", '*')` returns the array `["hello", "fellow", "students"]`.
Param. #1	The string to split.
Param. #2	The delimiter character.
Return value	The array of "fresh" strings result of the split.
Libc functions	`malloc(3)`, `free(3)`

ft_itoa	
Prototype	`char * ft_itoa(int n);`
Description	Allocate (with `malloc(3)`) and returns a "fresh" string ending with `'\0'` representing the integer `n` given as argument. Negative numbers must be supported. If the allocation fails, the function returns `NULL`.
Param. #1	The integer to be transformed into a string.
Return value	The string representing the integer passed as argument.
Libc functions	`malloc(3)`

ft_putchar	
Prototype	`void ft_putchar(char c);`
Description	Outputs the character `c` to the standard output.
Param. #1	The character to output.
Return value	None.
Libc functions	`write(2)`.

ft_putstr	
Prototype	`void ft_putstr(char const *s);`
Description	Outputs the string `s` to the standard output.
Param. #1	The string to output.
Return value	None.
Libc functions	`write(2)`.

ft_putendl	
Prototype	`void ft_putendl(char const *s);`
Description	Outputs the string `s` to the standard output followed by a `'\n'`.
Param. #1	The string to output.
Return value	None.
Libc functions	`write(2)`.

ft_putnbr	
Prototype	`void ft_putnbr(int n);`
Description	Outputs the integer `n` to the standard output.
Param. #1	The integer to output.
Return value	None.
Libc functions	`write(2)`.

ft_putchar_fd	
Prototype	`void ft_putchar_fd(char c, int fd);`
Description	Outputs the char `c` to the file descriptor `fd`.
Param. #1	The character to output.
Param. #2	The file descriptor.
Return value	None.
Libc functions	`write(2)`.

ft_putstr_fd	
Prototype	`void ft_putstr_fd(char const *s, int fd);`
Description	Outputs the string `s` to the file descriptor `fd`.
Param. #1	The string to output.
Param. #2	The file descriptor.
Return value	None.
Libc functions	`write(2)`.

ft_putendl_fd	
Prototype	`void ft_putendl_fd(char const *s, int fd);`
Description	Outputs the string `s` to the file descriptor `fd` followed by a `'\n'`.
Param. #1	The string to output.
Param. #2	The file descriptor.
Return value	None.
Libc functions	`write(2)`.

ft_putnbr_fd	
Prototype	`void ft_putnbr_fd(int n, int fd);`
Description	Outputs the integer `n` to the file descriptor `fd`.
Param. #1	The integer to print.
Param. #2	The file descriptor.
Return value	None.
Libc functions	`write(2)`.