



Libft

Biblioteca ta proprie

Sumar: Acest proiect are ca obiectiv sa scrieti o biblioteca de functii uzuale pe care le veti putea folosi in proiectele voastre.

Cuprins

I	Preambul	2
II	Introducere	3
III	Obiective	4
IV	Instructiuni generale	5
V	Parte obligatorie	7
V.1	Consideratii tehnice	7
V.2	Partea 1 - Functii ale libc	8
V.3	Partea 2 - Functii suplimentare	10
VI	Bonus	17
VII	Livrare si peer-evaluare	20

Capitolul I

Preambul

Acest prim proiect marcheaza debutul vostru in formarea ca programator. Profitati si cititi: [acest articol](#) si invatati incepand de azi ca tipurile diferentiaza un programator de o fiara. Daca nu intelegeti tot nu e grav. Asta vine cu timpul.

Pentru a va acompania muzical pe toata perioada acestui proiect va propunem o lista de formatii demne de interes. Daca nu va place insemna ca aveti gusturi indoielnice in materie, dar probabil ca aveti alte calitati cum ar fi sa aveti numerosi prieteni pe Facebook sau sa puteti sa va atingeti cotul cu limba. Pe scurt, formatiile sunt listate intr-o ordine arbitrara si nu e exhaustiva. Aceasta va este data cu titlu de exemplu si sunteti incurajati sa explorati bogata lor discografie.

- [Between The Buried And Me](#)
- [Between The Buried And Me, c'est bon, mangez-en](#)
- [Tesseract](#)
- [Chimp Spanner](#)
- [Emancipator](#)
- [Cynic](#)
- [Kalisia](#)
- [O.S.I](#)
- [Dream Theater](#)
- [Pain Of Salvation](#)
- [Crucified Barbara](#)

Capitolul II

Introducere

Proiectul `libft` reia conceptul din ziua 06 a piscinei, ca sa faceti o biblioteca de functii utile pe care sa le puteti apoi folosi in majoritatea proiectelor voastre de C din acest an si pentru a castiga mai mult timp. Acest proiect va solicita sa scrieti mult cod pe care l-ati scris deja pe parcursul piscinei, lucru ce reprezinta un bun prilej sa reluati aceasta tema, si sa vedeti cum ati avansat.



Figura II.1: Representarea Libft-ului vostru (viziune de artist)

Capitolul III

Obiective

Programarea în C este o activitate foarte laborioasă dacă nu avem acces la toate aceste mici funcții uzuale și foarte utile. De aceea vă propunem, prin acest proiect să alocăm timp să rescrieți aceste funcții, să le înțelegeți și să vi le apropiați. În acest fel veți putea reutiliza biblioteca voastră pentru a munci în mod eficient pe proiectele voastre următoare în C.

Acest proiect este pentru voi și o ocazie de a extinde lista funcțiilor cerute cu funcțiile voastre proprii și în acest fel de a face biblioteca voastră și mai utilă. Nu ezitați să completați biblioteca voastră `libft` pe tot timpul școlarității voastre, în momentul în care acest proiect nu va mai fi decât o amintire pentru voi.

Capitolul IV

Instructiuni generale

- Functiile se pot realiza in ordinea dorita si sunteti incurajati sa le folositi pe cele deja create pentru realizarea celorlalte. Dificultatea nu este crescatoare si ordinea subiectelor est arbitrara. E ca intr-un joc video in care puteti realiza misiunile in ordinea pe care o doriti si in care puteti utiliza castigurile din misiunile trecute pentru a le facilita pe urmatoarele.
- Proiectul votru trebuie sa fie la Norma
- In nici un caz, nu trebuie sa se opreasca in mod neasteptat (Segmentation fault, bus error, double free, etc.) in afara comportamentelor nedeterminate. Proiectul vostru ar fi considerat ca si nefunctional si ar primi nota 0 la sustinere.
- Orice memorie alocata trebuie eliberata in mod clar atunci când e necesar.
- Trebuie sa trimiteti, la radacina repository-ului vostru, un fisier numit **auteur** si continând login-ul vostru urmat de un '\n':

```
$>cat -e auteur  
xlogin$
```

- Va trebui sa trimiteti un fisier **C** pentru fiecare functie ce o aveti de realizat si un fisier **libft.h** care va contine toate prototipurile lor dar si fisierele **macros** si **typedefs** de care ati putea sa mai aveti nevoie. Toate aceste fisiere vor trebui sa se gaseasca la radacina repository-ului vostru.
- Va trebui sa trimiteti un **Makefile** care va compila sursele voastre spre o biblioteca statica numita **libft.a**.
- **Makefile** trebuie sa propuna cel putin regulile **\$(NAME)**, **all**, **clean**, **fclean** si **re** in ordinea care vi se pare cea mai adaptata.
- **Makefile**-ul vostru trebuie sa compileze munca voastra cu flag-urile de compilare **-Wall**, **-Wextra** si **-Werror**.
- Doar functiile urmatoare din **libc** sunt autorizate: **malloc(3)**, **free(3)** si **write(2)** iar utilizarea lor este restrânsa, vedeci mai jos.

- Trebuie bineînțeles sa includeti `include`-ul system necesar pentru utilizarea uneia sau alteia din cele 3 functii autorizate in fisierul vostru `.c`. Singurul `include` system pe care mai sunteti autorizati sa-l utilizati în plus este `string.h` pentru ca sa puteti avea acces la constanta `NULL` si la tipul `size_t`. Restul este interzis.
- Va încurajam sa realizati unul sai mai multe programe de test pentru biblioteca voastra. Chiar daca acestea **nu sunt obligatoriu de trimis pe repository si nu vor fi evaluate**, va vor permite sa testati usor munca voastra si a celorlalti. In afara piscinei la distanta care nu contine astfel de teste, veti gasi o mare utilitate a acestor teste in momentul corectarilor. Sunteti, in acest cadru, liberi de a utiliza testele voastre sau cele ale corectatorului/corectatului sau ambele iar logistica care sta la baza est la discretia voastra.

Capitolul V

Parte obligatorie

V.1 Consideratii tehnice

- Fisierul `libft.h` poate contine `macros` si `typedefs` dupa caz si nevoie.
- Un sir de caractere este **INTOTDEAUNA** terminat cu un `'\0'`, chiar daca acest lucru a fost omis in descrierea unei functii In caz contrar, daca trebuie sa lipseasca, acest lucru va fi indicat in mod explicit.
- Este interzisa utilizarea variabilelor globale.
- Daca aveti nevoie de functii auxiliare pentru a scrie functii complexe, trebuie sa definiti aceste funtii auxiliare in `static` respectând Norma.



Un inceput pentru a cunoaste ce este o functie statica (in engleza) :

<http://codingfreak.blogspot.com/2010/06/static-functions-in-c.html>

- Trebuie sa fiti atenti la type-urile utilizate si folositi in mod judicios cast-urile când este necesar, în mod special când un type `void *` este implicat. In general, evitati cast-urile implicite oricare ar fi tipurile implicate. Exemplu :

```
char    *str;

str = malloc(42 * sizeof(*str));      /* Wrong ! Malloc returneaza un void * (cast implicit) */
str = (char *) malloc(42 * sizeof(*str)); /* Right ! (cast explicit) */
```


V.2 Partea 1 - Functii ale libc

În această primă parte va trebui să rescrieți un ansamblu de funcții ale bibliotecii `libc` precum sunt descrise în `man`-ul lor respectiv pe sistemul vostru. Funcțiile trebuie să aibă exact același prototip și același comportament ca cele originale. Numele lor trebuie să aibă prefixul `ft_`. De exemplu `strlen` devine `ft_strlen`.



Anumite prototipuri ale funcțiilor pe care le veți rescrie folosesc calificativul de tip `"restrict"`. Acest cuvânt cheie face parte din standardul `c99`; trebuie deci să nu-l puneți în prototipul funcției și să nu compilați cu flag-ul `-std=c99`.

Va trebui să rescrieți următoarele funcții:

- `memset`
- `bzero`
- `memcpy`
- `memccpy`
- `memmove`
- `memchr`
- `memcmp`
- `strlen`
- `strdup`
- `strcpy`
- `strncpy`
- `strcat`
- `strncat`
- `strlcat`
- `strchr`
- `strrchr`
- `strstr`
- `strnstr`
- `strcmp`
- `strncmp`
- `atoi`
- `isalpha`

- isdigit
- isalnum
- isascii
- isprint
- toupper
- tolower

V.3 Partea 2 - Functii suplimentare

In aceasta parte trebuie sa scrieti un anumit numar de functii ce nu figureaza in `libc` sau se gasesc sub o forma diferita. Unele dintre aceste functii pot fi de interes pentru a facilita scrierea functiilor din prima parte.

ft_memalloc	
Prototip	<code>void * ft_memalloc(size_t size);</code>
Descriere	Alocare dinamica (cu <code>malloc(3)</code>) si returneaza o noua zona de memorie. Memoria alocata este initializata la 0. In cazul in care alocarea nu reuseste, functia returneaza <code>NULL</code> .
Param. #1	Dimensiunea zonei de memorie alocata.
Valoare de retur	Adresa zonei de memorie alocata.
Functii libc	<code>malloc(3)</code>

ft_memdel	
Prototip	<code>void ft_memdel(void **ap);</code>
Descriere	Ia ca parametru adresa unui pointer, deci zona de memorie alocata dinamic trebuie sa fie eliberata cu <code>free(3)</code> , la sfarsitul functiei pointerul trebuie sa fie initializat la <code>NULL</code> .
Param. #1	Adresa pointerului la care trebuie sa eliberam memoria si sa initializam la <code>NULL</code> .
Valoare de retur	Nimic.
Fonctii libc	<code>free(3)</code> .

ft_strnew	
Prototip	<code>char * ft_strnew(size_t size);</code>
Descriere	Alocati (cu <code>malloc(3)</code>) returnati un nou sir de caractere terminat cu un <code>'\0'</code> . Fiecare caracter din sir este initializat la <code>'\0'</code> . In cazul in care alocarea nu reuseste, functia returneaza <code>NULL</code> .
Param. #1	Dimensiunea sirului de caractere de alocat.
Valoare de retur	Sirul de caractere alocat si inializat la 0.
Functii libc	<code>malloc(3)</code>

ft_strdel	
Prototip	<code>void ft_strdel(char **as);</code>
Descriere	la ca parametru adresa unui sir de caractere care trebuie sa fie eliberat cu <code>free(3)</code> ; la sfarsitul functiei pointerul trebuie sa fie initializat la <code>NULL</code> .
Param. #1	Adresa pointerului spre zona de memorie cetrebuie eliberata si initializarea sa la valoarea <code>NULL</code> .
Valoare de retur	Nimic.
Functii libc	<code>Free(3)</code> .

ft_strclr	
Prototip	<code>void ft_strclr(char *s);</code>
Descriere	Atribuie valoarea <code>'\0'</code> la toate caracterele din sirul dat in parametru.
Param. #1	Sirul de caractere de sters.
Valoare de retur	Nimic.
Functii libc	Niciuna.

ft_striter	
Prototip	<code>void ft_striter(char *s, void (*f)(char *));</code>
Descriere	Aplica functia <code>f</code> la fiecare caracter din sirul de caractere trimis ca parametru. Adresa fiecarui caracter este trimisa ca parametru la functia <code>f</code> pentru a putea sa o modifice in caz de nevoie.
Param. #1	Sirul de caractere de iterat.
Param. #2	Functia de apelat asupra fiecarui caracter al sirului <code>s</code> .
Valoare de retur	Nimic.
Functii libc	Niciuna.

ft_striteri	
Prototip	<code>void ft_striteri(char *s, void (*f)(unsigned int, char *));</code>
Descriere	Aplica functia <code>f</code> la fiecare caracter din sirul de caractere trimis ca parametru. Adresa fiecarui caracter este trimisa ca al doilea parametru, precizand indexul ca prim parametru la functia <code>f</code> pentru a putea sa o modifice in caz de nevoie.
Param. #1	Sirul de caractere de iterat.
Param. #2	Functia de apelat asupra fiecarui caracter al sirului <code>s</code> si pe indexul sau.
Valoare de retur	Nimic.
Functii libc	Niciuna.

ft_strmap	
Prototip	<code>char * ft_strmap(char const *s, char (*f)(char));</code>
Descriere	Aplica functia <code>f</code> la fiecare caracter din sirul de caractere dat ca parametru, pentru a crea un nou sir (cu <code>malloc(3)</code>) care va avea in fiecare caracter din sir, caracterul rezultat primit de la apelul functiei <code>f</code> asupra caracterului cu acelasi index din sir.
Param. #1	Sirul de caractere de iterat.
Param. #2	Functia de apelat asupra fiecarui caracter al sirului <code>s</code> .
Valoare de retur	Sirul nou rezultand de la aplicarea succesiva a functiei <code>f</code> .
Functii libc	<code>malloc(3)</code>

ft_strmapi	
Prototip	<code>char * ft_strmapi(char const *s, char (*f)(unsigned int, char));</code>
Descriere	Aplica functia <code>f</code> precizand indexul la fiecare caracter din sirul de caractere dat in parametru, pentru a crea un nou sir (cu <code>malloc(3)</code>) care va avea in fiecare caracter din sir, caracterul rezultat primit de la apelul functiei <code>f</code> asupra caracterului cu acelasi index din sir.
Param. #1	Sirul de caractere asupra caruia se face iteratia.
Param. #2	Functia de apelat asupra fiecarui caracter <code>s</code> precizand indexul sau.
Valoare de retur	Noul sir rezultat in urma aplicarilor succesive ale functiei <code>f</code> .
Functii libc	<code>malloc(3)</code>

ft_strequ	
Prototip	<code>int ft_strequ(char const *s1, char const *s2);</code>
Descriere	Compara lexical <code>s1</code> si <code>s2</code> . Daca aceste doua siruri sunt egale functia va returna 1, daca nu 0.
Param. #1	Primul sir de comparat.
Param. #2	Al doilea sir de comparat.
Valoare de retur	1 sau 0 In functie de siruri daca sunt egale sau nu.
Functii libc	Niciuna.

ft_strnequ	
Prototip	<code>int ft_strnequ(char const *s1, char const *s2, size_t n);</code>
Descriere	Compara lexical <code>s1</code> si <code>s2</code> pana la <code>n</code> caractere sau pana cand un <code>'\0'</code> a fost gasit. Daca aceste doua siruri sunt egale functia va returna 1, daca nu 0.
Param. #1	Primul sir de comparat.
Param. #2	Al doilea sir de comparat.
Param. #3	Numarul maxim de caractere de comparat.
Valoare de retur	1 sau 0 In functie de siruri daca sunt egale sau nu.
Functii libc	Niciuna.

ft_strsub	
Prototip	<code>char * ft_strsub(char const *s, unsigned int start, size_t len);</code>
Descriere	Aloca (cu <code>malloc(3)</code>) si returneaza o copie de la o parte din sirul trimis ca parametru. Copia va incepe la indexul <code>start</code> si are ca lungime <code>len</code> . Daca <code>start</code> si <code>len</code> nu delimiteaza un sir valid, comportamentul este nedeterminat. In cazul in care alocarea nu reuseste, functia returneaza <code>NULL</code> .
Param. #1	Sirul de caractere unde trebuie sa cautati partea de copiat.
Param. #2	Indexul de unde incepem sa copiem sirul.
Param. #3	Lungimea partii de sir de copiat.
Valoare de retur	Sirul copiat.
Functii libc	<code>malloc(3)</code>

ft_strjoin	
Prototip	<code>char * ft_strjoin(char const *s1, char const *s2);</code>
Descriere	Aloca (cu <code>malloc(3)</code>) si returneaza un sir de caractere nou, delimitandu-i sfarsitul cu <code>'\0'</code> acest sir va contine sirurile concatenate <code>s1</code> si <code>s2</code> . Daca alocarea nu reuseste, functia va returna <code>NULL</code> .
Param. #1	Sirul de caractere prefix.
Param. #2	Sirul de caractere postfix.
Valoare de retur	Noul sir resultand din concatenarea sirului <code>s1</code> cu <code>s2</code> .
Functii libc	<code>malloc(3)</code>

ft_strtrim	
Prototip	<code>char * ft_strtrim(char const *s);</code>
Descriere	Aloca (cu <code>malloc(3)</code>) si returneaza o copie a sirului transmis in parametru, fara spatii albe "whitespaces" la inceput si la sfarsitul sirului. Consideram ca spatii albe caracterele urmatoare: ' ', '\n' si '\t'. Daca <code>s</code> nu contine spatii albe, la inceputul si la sfarsitul sirului, functia va transmite o copie a sirului <code>s</code> . Daca alocarea nu reuseste, functia va returna <code>NULL</code> .
Param. #1	Sirul de caractere ce trebuie decupat.
Valoare de retur	Noul sir de caractere decupat sau o copie al sirului <code>s</code> .
Functii libc	<code>malloc(3)</code>

ft_strsplit	
Prototip	<code>char ** ft_strsplit(char const *s, char c);</code>
Descriere	Aloca (cu <code>malloc(3)</code>) si returneaza un nou tablou de caractere (toate sirurile se vor termina cu un '\0', deci si tabloul) care desparte in cuvinte un sir de caractere <code>s</code> in functie de un alt caracter <code>c</code> . Daca alocarea nu reuseste, functia va returna <code>NULL</code> . Exemplu: <code>ft_strsplit("salut*les***etudiants*", '*')</code> trimite tabloul ["salut", "les", "etudiants"].
Param. #1	Sirul de despartit in cuvinte.
Param. #2	Caracterul delimitator.
Valoare de retur	Noul tablou de siruri de caractere rezultand din decuparea sirului <code>s</code> .
Functii libc	<code>malloc(3)</code>

ft_itoa	
Prototip	<code>char * ft_itoa(int n);</code>
Descriere	Aloca (cu <code>malloc(3)</code>) si returneaza un nou sir de caractere care se va termina cu un '\0' reprezentand intregul <code>n</code> trimis ca parametru. Trebuie gestionate si numerele negative. Daca alocarea nu reuseste, functia va returna <code>NULL</code> .
Param. #1	Intregul care trebuie convertit intr-un sir de caractere.
Valoare de retur	Sirul de caractere care reprezinta intregul transmis ca parametru.
Functii libc	<code>malloc(3)</code>

ft_putchar	
Prototip	<code>void ft_putchar(char c);</code>
Descriere	Afiseaza caracterul <code>c</code> la iesirea standard.
Param. #1	Caracterul de afisat.
Valoare de retur	Niciuna.
Functii libc	<code>write(2)</code> .

ft_putstr	
Prototip	<code>void ft_putstr(char const *s);</code>
Descriere	Afiseaza sirul de caractere <code>s</code> la iesirea standard.
Param. #1	Sirul de caractere de afisat.
Valoare de retur	Niciuna.
Functii libc	<code>write(2)</code> .

ft_putendl	
Prototip	<code>void ft_putendl(char const *s);</code>
Descriere	Afiseaza sirul de caractere <code>s</code> la iesirea standard urmat de un <code>'\n'</code> .
Param. #1	Sirul de caractere de afisat.
Valoare de retur	Niciuna.
Fonctii libc	<code>write(2)</code> .

ft_putnbr	
Prototip	<code>void ft_putnbr(int n);</code>
Descriere	Afiseaza intregul <code>n</code> la iesirea standard.
Param. #1	Intregul de afisat.
Valoare de retur	Niciuna.
Functii libc	<code>write(2)</code> .

ft_putchar_fd	
Prototip	<code>void ft_putchar_fd(char c, int fd);</code>
Descriere	Scrie caracterul <code>c</code> intr-un descriptor de fisier <code>fd</code> .
Param. #1	Caracterul de scris.
Valoare de retur	Niciuna.
Functii libc	<code>write(2)</code> .

ft_putstr_fd	
Prototip	<code>void ft_putstr_fd(char const *s, int fd);</code>
Descriere	Scrie sirul de caractere <code>s</code> intr-un descriptor de fisier <code>fd</code> .
Param. #1	Sirul de caractere de scris.
Valoare de retur	Niciuna.
Fonctii libc	<code>write(2)</code> .

ft_putendl_fd	
Prototip	<code>void ft_putendl_fd(char const *s, int fd);</code>
Descriere	Scrie sirul de caractere <code>s</code> intr-un descriptor de fisier <code>fd</code> urmat de un <code>'\n'</code> .
Param. #1	Sirul de caractere de scris.
Valoare de retur	Niciuna.
Fonctii libc	<code>write(2)</code> .

ft_putnbr_fd	
Prototip	<code>void ft_putnbr_fd(int n, int fd);</code>
Descriere	Scrie intregul <code>n</code> intr-un descriptor de fisier <code>fd</code> .
Param. #1	Integul de scris.
Valoare de retur	Niciuna.
Fonctii libc	<code>write(2)</code> .

Capitolul VI

Bonus

Daca ati reusit perfect partea obligatorie, in aceasta sectiune va propunem un mod pentru a avansa mai departe. Exact cum ai cumpara un pachet de extensie la un joc video. Bonusurile vor fi luante in considerare doar daca obtineti cel putin un scor de 18/20 la partea obligatorie.

Daca ai niste functii de gestiunea memoriei si a sirurilor de caractere este foarte convenabil, dar va veti da rapid seama ca daca aveti si niste functii de de procesare a listelor ar fi si mai convenabil.

Veti utiliza urmatoarea structura pentru a reprezenta nodurile listei. Aceasta structura trebuie sa fie adaugata in fisierul vostru `libft.h`.

```
typedef struct    s_list
{
    void          *content;
    size_t        content_size;
    struct s_list *next;
} t_list;
```

Descrierea campurilor de la structura `t_list` este urmatoarea:

- **content:** Datele sunnt continute intr-un nod. Tipul `void *` permite stocarea oricarui tip de variabila.
- **content_size:** Dimensiunea datelor stocate. Tipul `void *` nu permite cunoasterea dimensiunii datelor vizate, deci necesita sa o salvam. Exemplu: sirul de caractere "42" are o dimensiune de 3 octeti si intregul 32 biti 42 are o dimensiune de 4 octeti.
- **next:** Adresa urmatorului nod din lista sau valoarea `NULL` daca este ultimul nod.

Urmatoarele functii va vor permite manipularea usoara a listelor.

ft_lstnew	
Prototip	<code>t_list * ft_lstnew(void const *content, size_t content_size);</code>
Descriere	Aloca (cu <code>malloc(3)</code>) si returneaza un nou nod. Campurile <code>content</code> si <code>content_size</code> din noul nod sunt initializate prin copia parametrului functiei. Daca parametrul <code>content</code> este nul, campul <code>content</code> este initializat la <code>NULL</code> si campul <code>content_size</code> este initializat la 0 ne tinand cont de valoarea parametrului <code>content_size</code> . Campul <code>next</code> este initializat la <code>NULL</code> . Daca alocarea nu reuseste, functia va returna <code>NULL</code> .
Param. #1	Continutul de adaugat in noul nod.
Param. #2	Dimensiunea continutului de adaugat la urmatorul nod.
Valoare de retur	Noul nod.
Functii libc	<code>malloc(3)</code>

ft_lstdelone	
Prototip	<code>void ft_lstdelone(t_list **alst, void (*del)(void *, size_t));</code>
Descriere	La ca parametru adresa la pointer la un nod si elibereaza memoria continutului acestui nod cu functia <code>del</code> transmisa ca parametru si apoi elibereaza memoria nodului cu functia <code>free(3)</code> . Memoria campului <code>next</code> nu trebuie in niciun caz sa fie eliberata. La sfarsit, functia va pune pointerul la nod la valoarea <code>NULL</code> (asemanator cu functia <code>ft_memdel</code> din partea obligatorie).
Param. #1	Adresa unui pointer la un nod de eliberat.
Valoare de retur	Niciuna.
Functii libc	<code>free(3)</code>

ft_lstdel	
Prototip	<code>void ft_lstdel(t_list **alst, void (*del)(void *, size_t));</code>
Descriere	La ca parametru adresa unui pointer la un nod si elibereaza memoria continutului acestui nod cu functia <code>del</code> transmisa ca parametru si apoi elibereaza memoria nodului cu functia <code>free(3)</code> . (acest procedeu se repeta pentru toti succesorii nodului primit ca parametru) Memoria campului <code>next</code> nu trebuie in niciun caz sa fie eliberata. La sfarsit, functia va pune pointerul la nod la valoarea <code>NULL</code> (asemanator cu functia <code>ft_memdel</code> din partea obligatorie).
Param. #1	Adresa pointerului la primul nod din lista de eliberat.
Valoare de retur	Niciuna.
Functii libc	<code>free(3)</code>

ft_lstadd	
Prototip	<code>void ft_lstadd(t_list **alst, t_list *new);</code>
Descriere	Adauga un element <code>new</code> in capul listei.
• Param. #1	Adresa unui pointer la primul element al listei.
Param. #2	Nodul de adaugat ca prim element al acestei liste.
Valoare de retur	Niciuna.
Functii libc	Niciuna.

ft_lstiter	
Prototip	<code>void ft_lstiter(t_list *lst, void (*f)(t_list *elem));</code>
Descriere	Parcurge lista <code>lst</code> aplicand la fiecare nod functia <code>f</code> .
• Param. #1	Pointerul la primul nod al unei liste.
Param. #2	Adresa unei functii ce va fi aplicata asupra tuturor nodurilor de la lista luata ca parametru.
Valoare de retur	Niciuna.
Fonctii libc	Niciuna.

ft_lstmap	
Prototip	<code>t_list * ft_lstmap(t_list *lst, t_list * (*f)(t_list *elem));</code>
Descriere	Parcurge lista <code>lst</code> si aplica fiecui nod functia <code>f</code> si creaza o noua lista cu ajutorul <code>malloc(3)</code> rezultand de la aplicarile succesive. Daca o alocare esueaza, functia returneaza <code>NULL</code> .
• Param. #1	Pointerul pe primul nod a unei liste.
Param. #2	Adresa unei functii ce va fi aplicata pe toate nodurile listei luata in parametru pentru a crea o noua lista.
Valoare de retur	Noua lista.
Fonctii libc	<code>malloc(3)</code>

Daca reusiti sa faceti perfect partea obligatorie si partea bonus, sunteti incurajati sa adaugati si alte functii care vi se par utile, pentru a mari biblioteca. Daca cel care va corecteaza le considera pertinente, puteti primi puncte suplimentare. Exemple: o versiune de `ft_strsplit` ce returneaza o lista de siruri in locul unui tablou de siruri; functia `ft_lstfold` similara functiei `reduce` din Python si functiei `List.fold_left` din OCaml (atentie la memory leak-uri!), functii de manipulare a tablourilor, stivelor, cozilor, maps, tabelor de dispersie (hash tables) etc. Limita este doar imaginatia voastra.

Capitolul VII

Livrare si peer-evaluare

Trebuie sa livrati, in repository-ul vostru **Git** ca de obicei. Doar continutul ce se afla in directorul vostru de lucru/livrare va fi evaluat.

Daca si numai daca realizati acest proiect in cadrul piscinei la distanta: Proiectul vostru va fi evaluat numai de catre moulineta.

Sinon: Dupa terminarea sustinerii, o moulineta va trece peste proiectul trimis. Nota finala va fi calculata tinând cont de notele primite la peer-evaluare si de nota de la moulineta.

Succes tuturor si nu uitati fisierul vostru **auteur** !