

C Ավազան C 12

<ակիրճ. այս փաստաթուղթը 42 դպրոցի C ավազանի C 12 մոդուլի նյութն t:

Ցանկ

I	Նախաբան		2
II	Ցուցումներ		4
Ш	Առաջադրանք	00 : ft_create_elem	6
IV	Առաջադրանք	01 : ft_list_push_front	7
V	Առաջադրանք	02 : ft_list_size	8
VI	Առաջադրանք	03 : ft_list_last	9
VII	Առաջադրանք	04 : ft_list_push_back	10
VIII	Առաջադրանք	05 : ft_list_push_strs	11
IX	Առաջադրանք	06 : ft_list_clear	12
X	Առաջադրանք	07 : ft_list_at	13
XI	Առաջադրանք	08 : ft_list_reverse	14
XII	Առաջադրանք	09 : ft_list_foreach	15
XIII	Առաջադրանք	10 : ft_list_foreach_if	16
XIV	Առաջադրանք	11 : ft_list_find	17
XV	Առաջադրանք	12 : ft_list_remove_if	18
XVI	Առաջադրանք	13 : ft_list_merge	19
XVII	Առաջադրանք	14 : ft_list_sort	20
XVIII	Առաջադրանք	15 : ft_list_reverse_fun	21
XIX	Առաջադրանք	16 : ft_sorted_list_insert	22
XX	Առաջադրանք	17 : ft_sorted_list_merge	23
XXI	Հանձնում և ո ն	նկերն ընկերոջը ստուգում	24

Գլուխ I Նախաբան

<u>ՉԳՈԻՇԱՑՈԻՄ</u> Մի՛ կարդացեք հաջորդ էջը

Ձեզ զգուշացրել էին։

- «Աստղային պատերազմներ»-ում Դարթ Վեյդերը Լուկի հայրն է։
- «Սովորական կասկածյալներ»-ում Վերբալը Քեյսեր Սոուզն է։
- «Մարտական ակումբ»-ում Տայլեր Դարդենն ու պատմողը նույն մարդն են։
- «Վեցերորդ զգայարան»-ում Բրյուս Վիլլիսը հենց սկզբից մահացած է։
- «Ուրիշներ»-ում տան բնակիչները ուրվականներ են և հակառակը։
- «Բեմբի»-ում Բեմբիի մայրը մահանում է։
- «The Village»-ում հրեշները քաղաքացիներն են և ֆիլմի իրադարձությունները տեղի են ունենում մեր ժամանակներում։
- «Հարի Φոթթեր»-ում Դամբլդորը մահանում է։
- «Կապիկների մոլորակ» ֆիլմի իրադարձությունները տեղի են ունենում երկրի վրա։
- «Գահերի խաղ»-ում Ռոբը և Ջոֆֆերի Բարաթեոնը մահանում են իրենց հարսանիքի օրը։
- «*Մթնշաղ»*-ում վամպիրները շողում են արևի տակ։
- «Աստղային դարպասներ ԱԳ-1», 1-ին եթերաշրջան, 18-րդ Իպիզոդում Օնիլը և Քարթերը Անտարկտիդայում են։
- «*Խավարի ասպետը։ Լեգենդի վերածնունդը*» ֆիլմում Միրանդա Թեյթր Թալիա Ալգույն է։
- «Սուպեր Մարիո Բրոս»-ում արքայադուսարը մի ուրիշ դղյակում է։

Գլուխ II Յուցումներ

- Այս էջը ձեր միակ ուղեցույցն է։ Պատվող խոսակցություններին ուշադրություն մի՛ դարձրեք։
- Չգուշացում. մինչ առաջադրանքները հանձնել նորից ստուգե՜ք նյութը։ Ցանկացած պահի այս փաստաթուղթը կարող է փոփոխվել։
- Ուշադրություն դարձրե՛ք ձեր ֆայլերի և պահոցների թույլտվություններին։
- Բոլոր առաջադրանքները կատարելիս անհրաժեշտ է հետևել հանձման ընթացակարգին։
- Ձեր առաջադրանքները կստուգվեն ձեր դասընկերների կողմից։
- Քացի դրանից, ձեր առաջադրանքները կստուգվեն և կգնահատվեն Moulinette կոչվող ծրագրով։
- Moulinette-ը գնահատելիս շատ բծախնդիր է ու խիստ։ Այն ամբողջովին ավտոմատացված է, և գնահատման հարցում նրա հետ անհնար է բանակցել։ Այսպիսով, տհաճ անակնկալներից խուսափելու համար առաջադրանքները պետք է կատարվեն հնարավորինս անթերի։
- Moulinette-ն այնքան էլ լայնախոհ չէ։ Այն չի էլ փորձի հասկանալ ձեր կոդը, եթե վերջինս չի համապատասխանում Norm-ին։
- Moulinette-ի աշխատանքը հիմնված է norminette կոչվող ծրագրի վրա, որը ստուգում է, թե արդյոք ձեր ֆայլերը համապատասխանում են Norm-ին։ Կարճ ասած, norminette-ի ստուգման թեստր չանցած աշխատանքը չի ընդունվի։
- Առաջադրանքները դասավորված են ըստ բարդության` ամենապարզից ամենաբարդը։ Հաջողությամբ կատարված բարդ առաջադրանքները հաշվի չեն առնվի եթե պարզ առաջադրանքներից որևէ մեկը լիարժեք չի աշխատում։
- Արգելված ֆունկցիաների կիրառումը համարվում է խարդախություն։ Խարդախությունը պատժվում են -42-ով, և այս գնահատականը քննարկման ենթակա չէ։

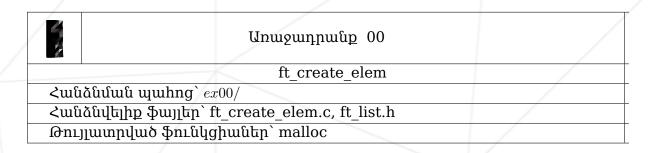
C Ավազան C 12

• main () ֆունկցիա պետք է հանձնել միայն այն դեպքում, եթե պահանջվի գրել ծրագիր։

- Moulinette-ը կոմպիլացվում է այս դրոշակների օգնությամբ` Wall Wextra -Werror, և գործածում է cc։
- Եթե ձեր ծրագիրը չկոմպիլացվի, կստանաք 0։
- Նյութում նշված ֆայլից բացի ձեր պահոցում <u>հավելյալ</u> ֆայլեր չպետք է լինեն։
- Հարցեր կա՞ն։ Դիմե՛ք աջ կողմում նստած դասընկերոջը։ Կամ էր դիմե՛ք ձախ կողմինին։
- Ձեր ուղեցույցներն են *Google-ը/ man-ը / համացանցը / ...*։
- Կարող եք օգտվել նաև ներքնացանցի ֆորումի "C Piscine" հատվածից կամ slack Piscine-ից։
- Մանրակրկիտ ուսումնասիրե՛ք օրինակները։ Շատ հնարավոր է, որ դրանք պահանջեն նյութում հստակորեն չնշված մանրամասներ։
- Հանու՜ն Օդինի, հանու՜ն Արանազդի` ուղեղներդ ի գո՜րծ
- <ետագա առաջադրանքների համար կօգտագործենք հետևյալ կառուցվածքը`

- Այս կառուցվածքը պետք է ներառել ft_list.h ֆայլում և հանձնել այն յուրաքանչյուր առաջադրանքի հետ։
- Հաշվի առե՛ք, որ առաջադրանք 01-ից սկսած՝ օգտագործելու ենք մեր ft_create_elem-ը (ցանկալի կլինի, որ ֆունկցիայի նախատիպը լինի ft_list.h ֆայլում...)։

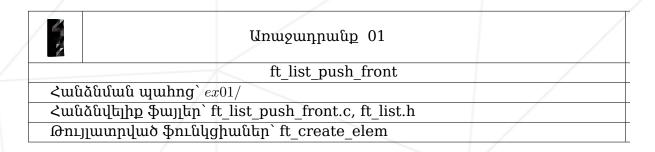
Գլուխ III Առաջադրանք 00 ։ ft_create_elem



- Ստեղծել ft_create_elem ֆունկցիան, որը ստեղծում է t_list տեսակի նոր տարր։
- Այն data-ն պետք է վերագրի տրված արգումենտին, իսկ next-ը՝ NULL-ին։
- Նախատիպը պետք է լինի այսպիսին`

Գլուխ IV

Առաջադրանք 01 ։ ft_list_push_front

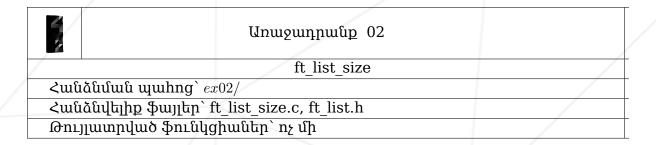


- Ստեղծել ft_list_push_front ֆունկցիան, որը t_list տեսակի նոր տարր է ավելացնում ցուցակի սկզբում։
- Այն պետք է data-ն վերագրի տրված արգումենտին։
- Հարկ եղած դեպքում այն կթարմացնի ցուցիչը ցուցակի սկզբում։
- Նախատիպը պետք է լինի այսպիսին`

void ft_list_push_front(t_list **begin_list, void *data);

Գլուխ V

Առաջադրանը 02 : ft_list_size

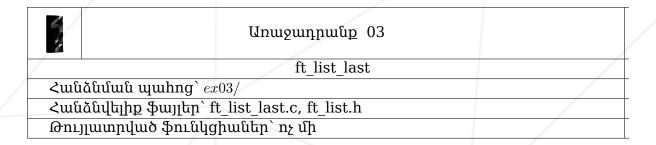


- Ստեղծել ft_list_size ֆունկցիան, որը վերադարձնում է ցուցակի տարրերի քանակը։
- Նախատիպը պետք է լինի այսպիսին՝

int ft_list_size(t_list *begin_list);

Գլուխ VI

Առաջադրանը 03 : ft_list_last

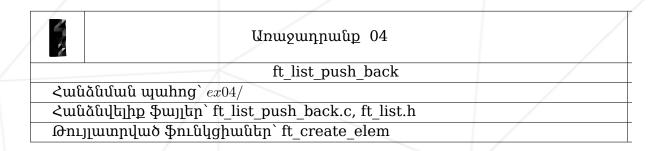


- Ստեղծել ft_list_last ֆունկցիան, որը վերադարձնում է ցուցակի վերջին տարրը։
- Նախատիպը պետք է լինի այսպիսին՝

t_list *ft_list_last(t_list *begin_list);

Գլուխ VII

Առաջադրանք 04 ։ ft_list_push_back

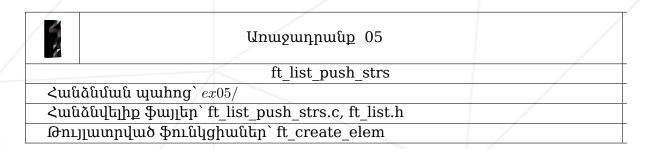


- Ստեղծել ft_list_push_back ֆունկցիան, որը t_list տեսակի տարր է ավելացնում ցուցակի վերջում։
- Այն պետք է data-ն վերագրի տրված արգումենտին։
- Հարկ եղած դեպքում այն կթարմացնի ցուցիչը ցուցակի սկզբում։
- Նախատիպը պետք է լինի այսպիսին`

void ft_list_push_back(t_list **begin_list, void *data);

Գլուխ VIII

Առաջադրանք 05 ։ ft_list_push_strs

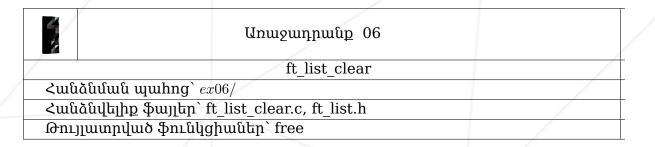


- Ստեղծել ft_list_push_strs ֆունկցիան, որը ստեղծում է մի նոր ցուցակ, որը ներառում է strs-ի տարրով ցուցանշված բոլոր տողերը։
- size-ր strs-ի չափն է։
- Առաջին տարրը պետք է լինի ցուցակի վերջում։
- Վերադարձվում է ցուցակի առաջին տարրի հղման հասցեն։
- Նախատիպը պետք է լինի այսպիսին`

t_list *ft_list_push_strs(int size, char **strs);

Գլուխ IX

Առաջադրանը 06 : ft_list_clear

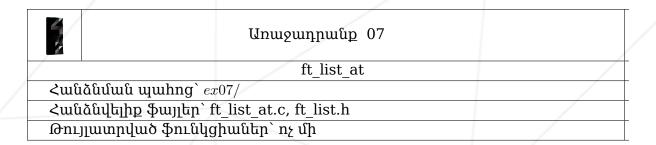


- Ստեղծել ft_list_clear ֆունկցիան, որը դատարկում և ջնջում է ցուցակի բոլոր հղումները։
- free_fct-ն դատարկում է յուրաքանչյուր data-ն։
- Նախատիպը պետք է լինի այսպիսին`

void ft_list_clear(t_list *begin_list, void (*free_fct)(void *));

Գլուխ X

Առաջադրանը 07 : ft_list_at

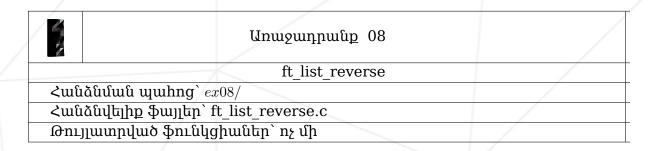


- Ստեղծել ft_list_at ֆունկցիան, որը վերադարձնում է ցուցակի Nրդ տարրը` հաշվի առնելով, որ ցուցակի առաջին տարրի դեպքում nbr-ը հավասար է 0։
- Սխալի դեպքում պետք է վերադարձվի ոսև ցուցիչ։
- Նախատիպը պետք է լինի այսպիսին`

t_list *ft_list_at(t_list *begin_list, unsigned int nbr);

Գլուխ XI

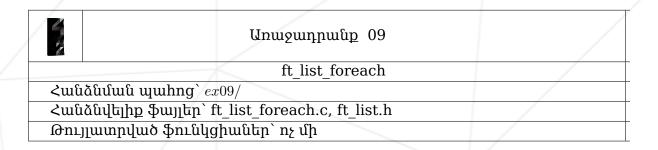
Առաջադրանք 08 ։ ft_list_reverse



- Ստեղծել ft_list_reverse ֆունկցիան, որը շրջում է ցուցակի տարրերի հերթականությունը։ Տարրերի արժեքները պետք է մնան անփոփոխ։
- Հաշվի առե՛ք, որ այդ ֆունկցիայում մենք օգտագործելու ենք մեր ft_list.h-ը։
- Նախատիպը պետք է լինի այսպիսին`

void ft_list_reverse(t_list **begin_list);

Գլուխ XII Առաջադրանք 09 ։ ft_list_foreach



- Ստեղծել ft_list_foreach ֆունկցիան, որը որպես արգումենտ փոխանցված ֆունկցիան կիրառում է ցուցակի բոլոր տարրերին։
- ք-ը պետք է կիրառել նույն հերթականությամբ, ինչ ցուցակում։
- Նախատիպը պետք է լինի այսպիսին`

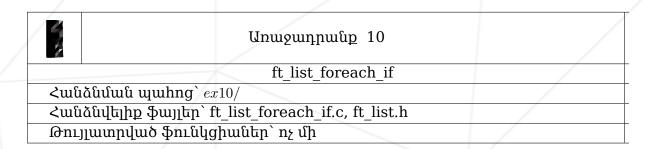
void ft_list_foreach(t_list *begin_list, void (*f)(void *));

• ƒ ցուցիչով տրված ֆունկցիան կգործածվի այսպես՝

(*f)(list_ptr->data);

Գլուխ XIII

Առաջադրանք 10 ։ ft_list_foreach_if



- Ստեղծել ft_list_foreach_if ֆունկցիան, որը որպես արգումենտ փոխանցված ֆունկցիան կիրառում է ցուցակի տարրերից մի քանիսի վրա։
- Ֆունկցիան կիրառել տարրերի վրա միայն այն դեպքում, երբ cmp-ն data_ref-ի հետ է, cmp-ն վերադարձնում է 0։
- f պետք է կիրառել նույն հերթականությամբ, ինչ ցուցակում։
- Նախատիպը պետք է լինի այսպիսին`

```
void ft_list_foreach_if(t_list *begin_list, void (*f)(void *), void
*data_ref, int (*cmp)())
```

• ք և շար ցուցիչներով տրված ֆունկցիաները կօգտագործվի այսպես՝

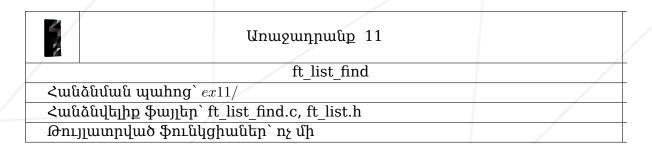
```
(*f)(list_ptr->data);
(*cmp)(list_ptr->data, data_ref);
```



Օրինակ՝ cmp ֆունկցիան կարող է լինել ft_strcmp...

Գլուխ XIV

Առաջադրանը 11 : ft_list_find



- Ստեղծել ft_list_find ֆունկցիան, որը վերադարձնում է առաջին տարրի արժեքի հասցեն, որի տվյալները data_ref-ի հետ համեմատելիս՝ օգտագործելով cmp, այնպես է անում, որ cmp-ն վերադարձնի 0։
- Նախատիպը պետք է լինի այսպիսին`

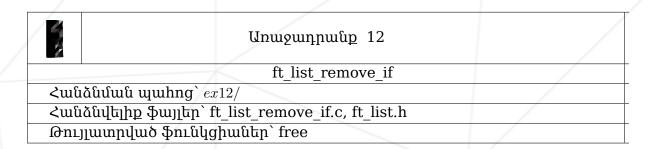
```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

• շար ցուցիչով տրված ֆունկցիան կօգտագործվի այսպես՝

(*cmp)(list_ptr->data, data_ref);

Գլուխ XV

Առաջադրանք 12 ։ ft_list_remove_if



- Ստեղծել ft_list_remove_if ֆունկցիան, որը ջնջում է բոլոր այն տարրերը, որոնց տվյալները` data_ref-ի հետ համեմատելիս՝ օգտագործելով cmp, այնպես է անում, որ cmp-ն վերադարձնի 0։
- Այն տարրի տվյալները, որը ջնջվել է, պետք է դատարկել free_fct-nվ։
- Նախատիպ<u>ը</u> պետք է լինի այսպիսին`

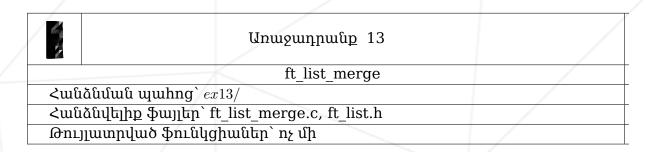
```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *);
```

• cmp և free_fct ցուցիչներով նշված ֆունկցիան կօգտագործվի այսպես՝

```
(*cmp)(list_ptr->data, data_ref);
(*free_fct)(list_ptr->data);
```

Գլուխ XVI

Առաջադրանք 13 ։ ft_list_merge

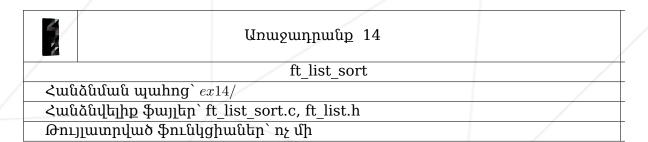


- Ստեղծել ft_list_merge ֆունկցիան, որը begin2 ցուցակի տարրերը դնում է մի ուրիշ՝ begin1 ցուցակի վերջում։
- Տարրերի ստեղծումը թույլատրված չէ։
- Նախատիպը պետք է լինի այսպիսին`

void ft_list_merge(t_list **begin_list1, t_list *begin_list2);

Գլուխ XVII

Uրաջադրանը 14 : ft_list_sort



- Ստեղծել ft_list_sort ֆունկցիան, որը ցուցակի տարրերը տեսակավորում է աճման կարգով` համեմատելով երկու տարրերը իրար հետ` օգտագործելով երկու տարրերի տվյալների համեմատման ֆունկցիա։
- Նախատիպը պետք է լինի այսպիսին`

```
void ft_list_sort(t_list **begin_list, int (*cmp)());
```

• cmp ցուցիչով նշված ֆունկցիան կօգտագործվի այսպես`

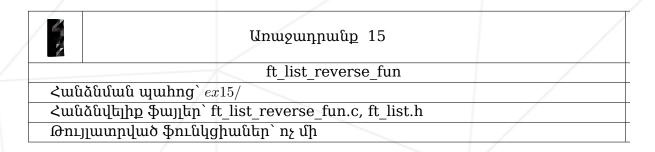
(*cmp)(list_ptr->data, list_other_ptr->data);



Օրինակ՝ այս cmp ֆունկցիան կարող է լինել ft_strcmp...

Գլուխ XVIII

Առաջադրանք 15 ։ ft_list_reverse_fun

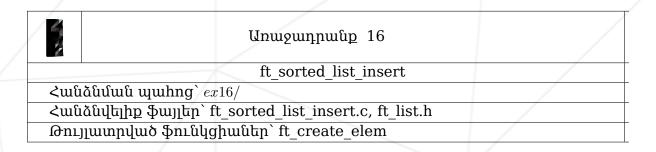


- Ստեղծել ft_list_reverse_fun ֆունկցիան, որը կշրջի ցուցակի տարրերի հերթականությունը։
- Նախատիպը պետք է լինի այսպիսին`

void ft_list_reverse_fun(t_list *begin_list);

Գլուխ XIX

Unugunpuup 16: ft_sorted_list_insert



- Ստեղծել ft_sorted_list_insert ֆունկցիան, որը ստեղծում է նոր տարր և այն ավելացնում է տեսակավորված ցուցակի մեջ այնպես, որ դրա աճման կարգը չի խախտվում։
- Նախատիպը պետք է լինի այսպիսին`

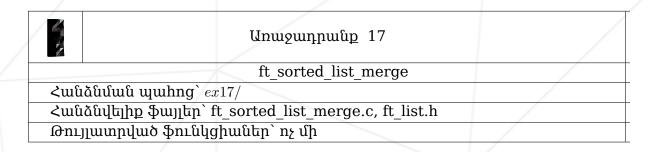
```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

• cmp ցուցիչով նշված ֆունկցիան կօգտագործվի այսպես`

(*cmp)(list_ptr->data, list_other_ptr->data);

Գլուխ XX

Առաջադրանք 17 : ft_sorted_list_merge



- Ստեղծել ft_sorted_list_merge ֆունկցիան, որը begin2 տեսակավորված ցուցակի տարրերը ավելացնում է begin1 տեսակավորված ցուցակում այնպես, որ begin1-ը մնում է աճման կարգով։
- Նախատիպը պետք է լինի այսպիսին`

```
void ft_sorted_list_merge(t_list **begin_list1, t_list *begin_list2, int (*cmp)());
```

• Այս շաթ ցուցիչով նշված ֆունկցիան կօգտագործվի այսպես՝

(*cmp)(list_ptr->data, list_other_ptr->data);

Գլուխ XXI

<անձնում և ընկերն ընկերոջը ստուգում

Հանձնեք ձեր առաջադրանքը Git պահոցում, ինչպես սովորաբար անում եք։ Ստուգման ժամանակ գնահատվելու է միայն ձեր պահոցի պարունակությունը։ Մի՛ վարանեք նորից ստուգել ձեր ֆայլերի անունները՝ համոզվելու համար, որ դրանք ճիշտ են։



Հարկավոր է հանձնել միայն այն ֆայլերը, որոնք պահանջվում են այս նախագիծը նկարագրող ֆայլում։