

C Piscine C 12

Summary: This document is the subject for the module C 12 of the C Piscine @ 42.

Version: 8

Contents

1	roreword	4
II	Instructions	4
III	Exercise 00 : ft_create_elem	6
IV	Exercise 01 : ft_list_push_front	7
\mathbf{V}	Exercise 02 : ft_list_size	8
VI	Exercise 03 : ft_list_last	9
VII	Exercise 04 : ft_list_push_back	10
VIII	Exercise 05 : ft_list_push_strs	11
IX	Exercise 06 : ft_list_clear	12
\mathbf{X}	Exercise 07 : ft_list_at	13
XI	Exercise 08 : ft_list_reverse	14
XII	Exercise 09 : ft_list_foreach	15
XIII	Exercise 10 : ft_list_foreach_if	16
XIV	Exercise 11 : ft_list_find	17
XV	Exercise 12 : ft_list_remove_if	18
XVI	Exercise 13 : ft_list_merge	19
XVII	Exercise 14 : ft_list_sort	20
XVIII	Exercise 15 : ft_list_reverse_fun	21
XIX	Exercise 16 : ft_sorted_list_insert	22
XX	Exercise 17 : ft_sorted_list_merge	23
XXI	Submission and peer-evaluation	24

Chapter I Foreword

SPOILER ALERT
DON'T READ THE NEXT PAGE

You've been warned.

- In Star Wars, Dark Vador is Luke's Father.
- In The Usual Suspects, Verbal is Keyser Soze.
- In Fight Club, Tyler Durden and the narrator are the same person.
- In The Sixth Sense, Bruce Willis has been dead since the beginning.
- In The others, the inhabitants of the house are ghosts and vice-versa.
- In Bambi, Bambi's mother dies.
- In The Village, monsters are the villagers and the movie actually takes place in our time.
- In Harry Potter, Dumbledore dies.
- In Planet of apes, the movie takes place on earth.
- In Game of thrones, Robb Stark and Joffrey Baratheon die on their wedding day.
- In Twilight, Vampires shine under the sun.
- In Stargate SG-1, Season 1, Episode 18, O'Neill and Carter are in Antartica.
- In The Dark Knight Rises, Miranda Tate is Talia Al'Gul.
- In Super Mario Bros, The princess is in another castle.

Chapter II

Instructions

- Only this page serves as your reference, do not trust rumors.
- Watch out! This document may change before submission.
- Ensure you have the appropriate permissions on your files and directories.
- You must follow the **submission procedures** for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- Additionally, your exercises will be evaluated by a program called **Moulinette**.
- Moulinette is meticulous and strict in its assessment. It is fully automated, and there is no way to negotiate with it. To avoid unpleasant surprises, be as thorough as possible.
- Moulinette is not open-minded. If your code does not adhere to the Norm, it won't attempt to understand it. Moulinette relies on a program called norminette to check if your files comply with the Norm. TL;DR: Submitting work that doesn't pass norminette's check makes no sense.
- These exercises are arranged in order of difficulty, from easiest to hardest. We will not consider a successfully completed harder exercise if an easier one is not fully functional.
- Using a forbidden function is considered cheating. Cheaters receive a grade of **-42**, which is non-negotiable.
- You only need to submit a **main()** function if we specifically ask for a **program**.
- Moulinette compiles with the following flags: -Wall -Wextra -Werror, using cc.
- If your program does not compile, you will receive a grade of **0**.
- You **cannot** leave **any** additional file in your directory beyond those specified in the assignment.
- Have a question? Ask the peer on your right. If not, try the peer on your left.

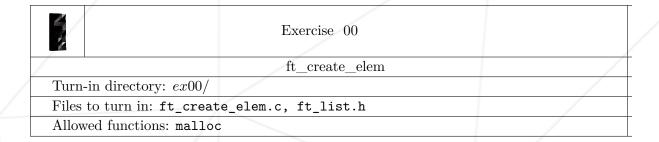
C Piscine

- Your reference guide is called Google / man / the Internet / ...
- Check the "C Piscine" section of the forum on the intranet or the Piscine on Slack.
- Carefully examine the examples. They may contain crucial details that are not explicitly stated in the assignment...
- By Odin, by Thor! Use your brain!!!
- For the following exercises, you have to use the following structure:

- You'll have to include this structure in a file ft_list.h and submit it for each exercise.
- From exercise 01 onward, we'll use our ft_create_elem, so make arrangements (it could be useful to have its prototype in a file ft_list.h...).

Chapter III

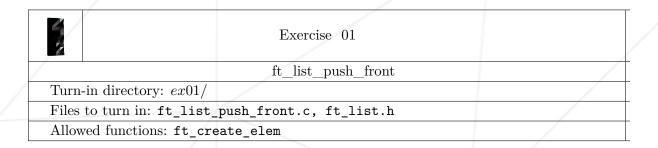
Exercise 00: ft_create_elem



- Create the function ft_create_elem, which creates a new element of t_list type.
- It should assign data to the given argument and next to NULL.
- Here is how it should be prototyped:

Chapter IV

Exercise 01: ft_list_push_front

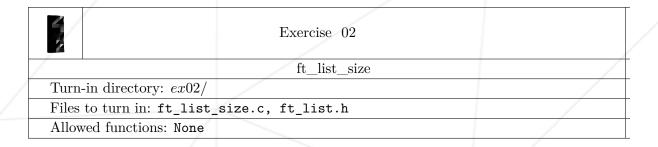


- Create the function ft_list_push_front, which adds a new element of type t_list to the beginning of the list.
- It should assign data to the given argument.
- If necessary, it will update the pointer at the beginning of the list.
- Here is how it should be prototyped:

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

Chapter V

Exercise 02 : ft_list_size

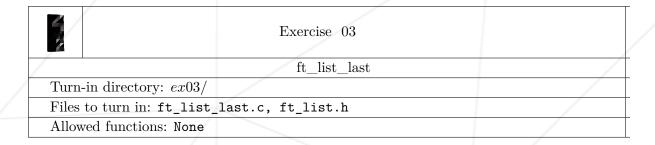


- Create the function ft_list_size, which returns the number of elements in the list.
- Here is how it should be prototyped:

int ft_list_size(t_list *begin_list);

Chapter VI

Exercise 03: ft_list_last

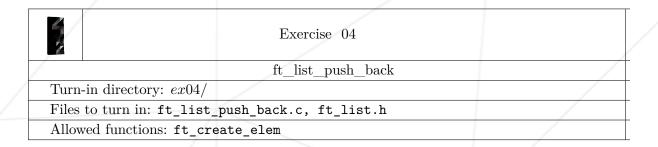


- Create the function ft_list_last, which returns the last element of the list.
- Here is how it should be prototyped:

t_list *ft_list_last(t_list *begin_list);

Chapter VII

Exercise 04: ft_list_push_back

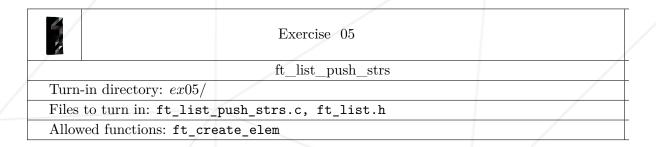


- Create the function ft_list_push_back, which adds a new element of t_list type at the end of the list.
- It should assign data to the given argument.
- If necessary, it will update the pointer at the beginning of the list.
- Here is how it should be prototyped:

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

Chapter VIII

Exercise 05: ft_list_push_strs

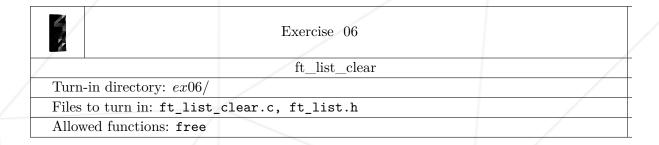


- Create the function ft_list_push_strs, which creates a new list that includes all the strings pointed to by the elements in strs.
- size is the size of strs.
- The first element should be at the end of the list.
- The first link's address in the list is returned.
- Here is how it should be prototyped:

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

Chapter IX

Exercise 06: ft_list_clear

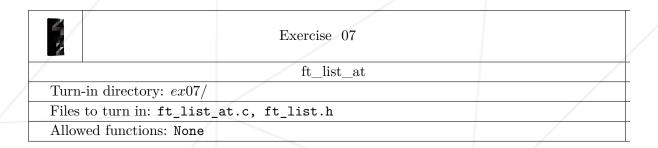


- Create the function ft_list_clear, which removes and frees all links from the list.
- free_fct is used to free each data.
- Here is how it should be prototyped:

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

Chapter X

Exercise 07: ft_list_at



- Create the function ft_list_at, which returns the Nth element of the list, knowing that the first element of the list is when nbr equals 0.
- In case of error, it should return a null pointer.
- Here is how it should be prototyped:

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

Chapter XI

Exercise 08 : ft_list_reverse

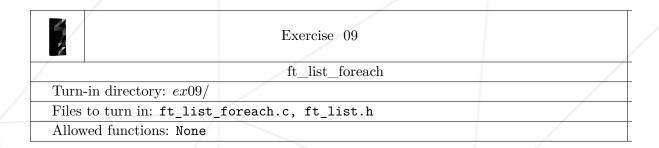
	Exercise 08	
/	ft_list_reverse	
Turn-in directory: $ex08/$		
Files to turn in: ft_list		
Allowed functions: None		

- Create the function ft_list_reverse, which reverses the order of a list's elements. The value of each element must remain the same.
- \bullet Beware that in this function, we will use our own ft_list.h.
- Here is how it should be prototyped:

void ft_list_reverse(t_list **begin_list);

Chapter XII

Exercise 09 : ft_list_foreach



- Create the function ft_list_foreach, which applies the function given as an argument to each of the list's elements.
- f should be applied in the same order as the list.
- Here is how it should be prototyped:

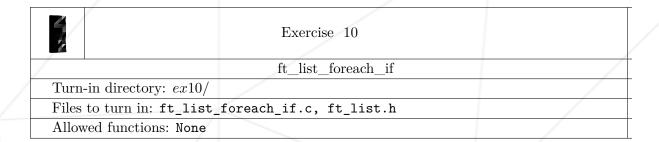
```
void ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

• The function pointed to by f will be used as follows:

(*f)(list_ptr->data);

Chapter XIII

Exercise 10: ft_list_foreach_if



- Create the function ft_list_foreach_if, which applies the function given as an argument to some of the list's elements.
- Only apply the function to the elements when cmp with data_ref returns 0.
- f should be applied in the same order as the list.
- Here is how it should be prototyped:

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

• Functions pointed to by f and by cmp will be used as follows:

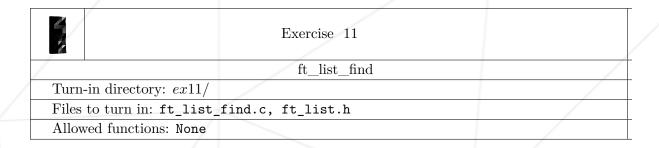
```
(*f)(list_ptr->data);
(*cmp)(list_ptr->data, data_ref);
```



For example, the function cmp could be ft_strcmp...

Chapter XIV

Exercise 11: ft_list_find



- Create the function ft_list_find which returns the address of the first element's data where comparing it to data_ref with cmp causes cmp to return 0.
- Here's how it should be prototyped:

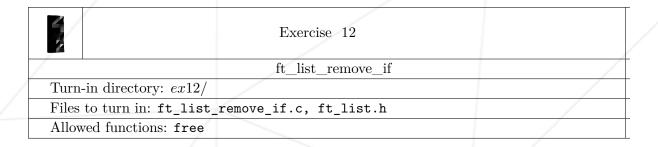
```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

• The function pointed to by cmp will be used as follows:

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

Chapter XV

Exercise 12: ft_list_remove_if



- Create the function ft_list_remove_if which removes from the list all elements whose data, when compared to data_ref using cmp, causes cmp to return 0.
- The data from an element to be erased should be freed using free_fct.
- Here's how it should be prototyped:

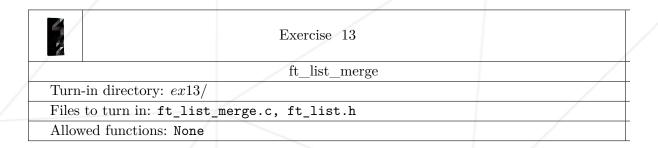
```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *);
```

• The functions pointed to by cmp and free_fct will be used as follows:

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

Chapter XVI

Exercise 13: ft_list_merge

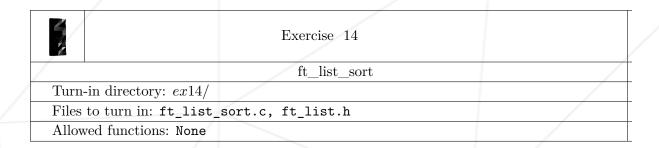


- Create the function ft_list_merge which places elements of a list begin2 at the end of another list begin1.
- Element creation is not authorised.
- Here's how it should be prototyped:

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

Chapter XVII

Exercise 14: ft_list_sort



- Create the function ft_list_sort which sorts the list's elements in ascending order by comparing two elements and their data using a comparison function.
- Here's how it should be prototyped:

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

• The function pointed to by cmp will be used as follows:

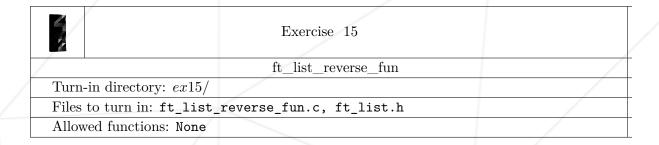
```
(*cmp)(list_ptr->data, list_other_ptr->data);
```



cmp could be for instance ft_strcmp.

Chapter XVIII

Exercise 15 : ft__list__reverse__fun

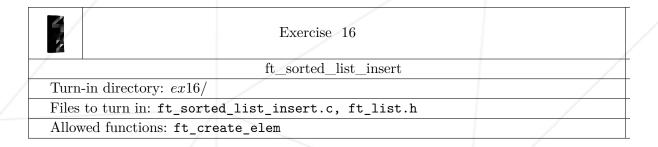


- Create the function ft_list_reverse_fun which reverses the order of the elements in the list.
- Here's how it should be prototyped:

void ft_list_reverse_fun(t_list *begin_list);

Chapter XIX

Exercise 16: ft_sorted_list_insert



- Create the function ft_sorted_list_insert which creates a new element and inserts it into a list sorted so that it remains sorted in ascending order.
- Here's how it should be prototyped:

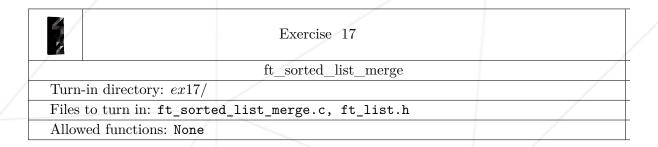
```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

• Function pointed by cmp will be used as follows:

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

Chapter XX

Exercise 17: ft_sorted_list_merge



- Create the function ft_sorted_list_merge which integrates the elements of a sorted list begin2 in another sorted list begin1, so that begin1 remains sorted by ascending order.
- Here's how it should be prototyped:

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

• Function pointed by cmp will be used as follows:

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

Chapter XXI

Submission and peer-evaluation

Submit your assignment to your Git repository as usual. Only the work inside your repository will be evaluated during the defense. Make sure to double-check the filenames to ensure they are correct.



You must submit only the files specified in the project instructions.