

C Piscine C 12

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

Contents

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

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 Sixth Sens, Bruce Willis is 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 Jo rey 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 will serve as reference: do not trust rumors.
- Watch out! This document could potentially change before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called norminette to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass norminette's check.
- These exercises are carefully laid out by order of diculty from easiest to hardest. We will not take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get −42, and this grade is non-negotiable.
- You'll only have to submit a main() function if we ask for a program.
- Moulinette compiles with these flags: -Wall -Wextra -Werror, and uses gcc.
- If your program doesn't compile, you'll get 0.
- You <u>cannot</u> leave <u>any</u> additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.

C Piscine

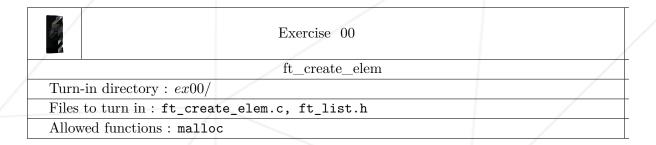
C 12

- Your reference guide is called Google / man / the Internet /
- Check out the "C Piscine" part of the forum on the intranet, or the slack Piscine.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- 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

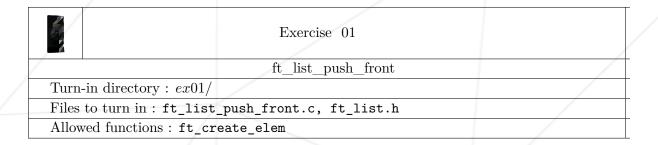
Exercice 00: ft_create_elem



- 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's how it should be prototyped :

Chapter IV

Exercice 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'll update the pointer at the beginning of the list.
- Here's how it should be prototyped :

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

Chapter V

Exercice 02 : ft_list_size

| | Exercise 02 | |
|--|--------------|--|
| | ft_list_size | |
| Turn-in directory : $ex02/$ | | |
| Files to turn in : ft_list_size.c, ft_list.h | | |
| Allowed functions : None | | |

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

int ft_list_size(t_list *begin_list);

Chapter VI

Exercice 03: ft_list_last

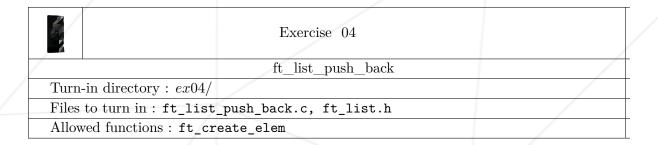
| | Exercise 03 | |
|--|--------------|--|
| | ft_list_last | |
| Turn-in directory : $ex03$ / | | |
| Files to turn in : ft_list_last.c, ft_list.h | | |
| Allowed functions : None | | |

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

t_list *ft_list_last(t_list *begin_list);

Chapter VII

Exercice 04: ft_list_push_back

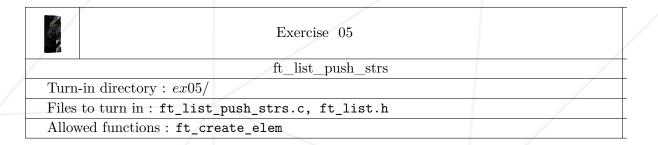


- Create the function ft_list_push_back which adds a new element of t_list type at the end of the list.
- It should assign data to the given argument.
- If necessary, it'll update the pointer at the beginning of the list.
- Here's how it should be prototyped :

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

Chapter VIII

Exercice 05: ft_list_push_strs



- Create the function ft_list_push_strs which creates a new list that includes all the string pointed by the element 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's how it should be prototyped :

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

Chapter IX

Exercice 06: ft_list_clear

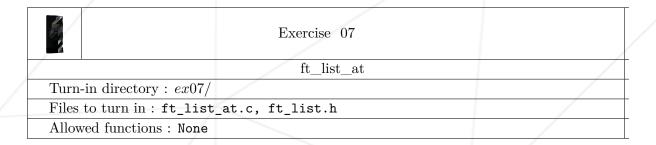
| | Exercise 06 | |
|---|-------------------|---|
| / | ft_list_clear | |
| Turn-in directory : $ex06/$ | | |
| Files to turn in : ft_list_clear.c, ft_list.h | | / |
| Allowed functions : free | | |

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

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

Chapter X

Exercice 07: ft_list_at

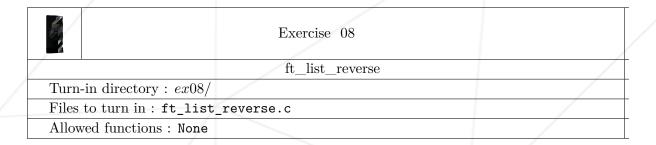


- Create the function ft_list_at which returns the Nth element of the list, knowing that the first element of the list is when nbr equal 0.
- In case of error, it should return a null pointer.
- Here's how it should be prototyped :

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

Chapter XI

Exercice 08 : ft_list_reverse

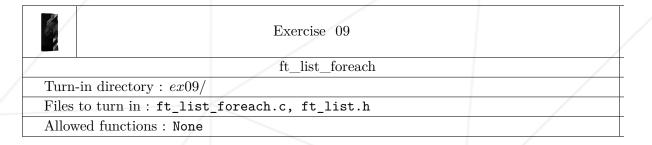


- Create the function ft_list_reverse which reverses the order of a list's elements. The value of each element must remain the same.
- Beware in that function we will use our own ft_list.h
- Here's how it should be prototyped :

void ft_list_reverse(t_list **begin_list);

Chapter XII

Exercice 09: ft_list_foreach



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

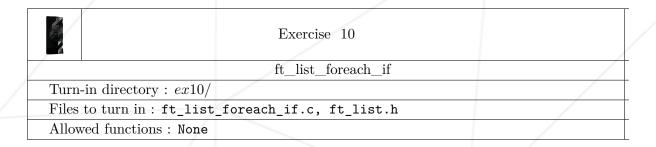
```
void ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

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

(*f)(list_ptr->data);

Chapter XIII

Exercice 10: ft_list_foreach_if



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

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

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

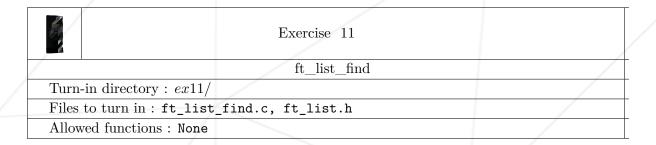
```
(*f)(list_ptr->data);
(*cmp)(list_ptr->data, data_ref);
```



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

Chapter XIV

Exercice 11: ft_list_find



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

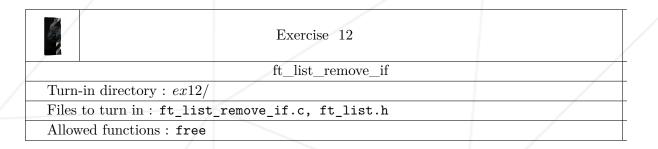
```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

• Function pointed by cmp will be used as follows:

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

Chapter XV

Exercice 12: ft_list_remove_if



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

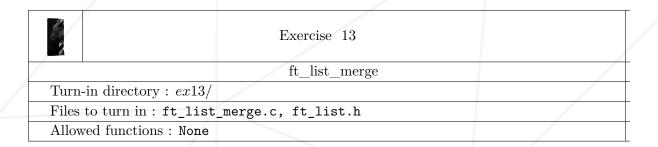
```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *)
```

• Function pointed by cmp and by free_fct will be used as follows :

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

Chapter XVI

Exercice 13: ft_list_merge

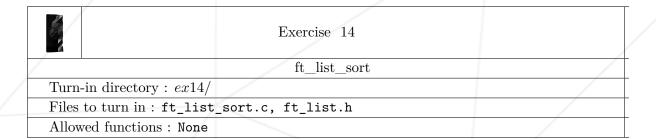


- Create the function ft_list_merge which places elements of a list begin2 at the end of an other list begin1.
- Element creation is not authorised.
- Here's how it should be prototyped :

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

Chapter XVII

Exercice 14: ft_list_sort



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

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

• Function pointed by cmp will be used as follows :

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



cmp could be for instance ft_strcmp.

Chapter XVIII

Exercice 15 : ft__list__reverse__fun

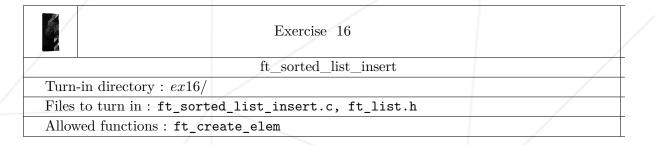
| | Exercise 15 | |
|--|-----------------------------|---|
| | ft_list_reverse_fun | |
| r | Furn-in directory : $ex15/$ | |
| Files to turn in: ft_list_reverse_fun.c, ft_list.h | | / |
| _ | Allowed functions : None | |

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

void ft_list_reverse_fun(t_list *begin_list);

Chapter XIX

Exercice 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 :

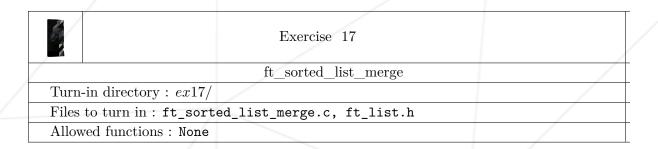
```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

• Function pointed by cmp will be used as follows:

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

Chapter XX

Exercice 17: ft_sorted_list_merge



- Create the function ft_sorted_list_merge which integrates the elements of a sorted list begin1 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);