Chapter III

Exercice 00 : ft_create_elem

	Exercice : 00
ft_create_elem	
Turn-in directory : $ex00/$	
Files to turn in : `ft_create_elem.c, ft_list.h`	
Allowed functions : `malloc`	
Remarks : `n/a`	

- 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 :

```
t_list      *ft_create_elem(void *data);
```

Chapter IV

Exercice 01 : ft_list_push_back

	Exercice : 01
ft_list_push_back	
Turn-in directory : $ex01/$	
Files to turn in : `ft_list_push_back.c, ft_list.h`	
Allowed functions : `ft_create_elem`	
Remarks : `n/a`	

- 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 V

Exercice 02 : ft_list_push_front

	Exercice : 02
ft_list_push_front	
Turn-in directory : $ex02/$	
Files to turn in : `ft_list_push_front.c, ft_list.h`	
Allowed functions : `ft_create_elem`	
Remarks : `n/a`	

- 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 VI

Exercice 03 : ft_list_size

	Exercice : 03
ft_list_size	
Turn-in directory : $ex03/$	
Files to turn in : `ft_list_size.c, ft_list.h`	
Allowed functions : `Nothing`	
Remarks : `n/a`	

- 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 VII

Exercice 04 : ft_list_last

	Exercice : 04
ft_list_last	
Turn-in directory : $ex04/$	
Files to turn in : `ft_list_last.c, ft_list.h`	
Allowed functions : `Nothing`	
Remarks : `n/a`	

- 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 VIII

Exercice 05 : ft_list_push_params

	Exercice : 05
ft_list_push_params	
Turn-in directory : $ex05/$	
Files to turn in : `ft_list_push_params.c, ft_list.h`	
Allowed functions : `ft_create_elem`	
Remarks : `n/a`	

- Create the function `ft_list_push_params` which creates a new list that includes command-line arguments.
- The first argument 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_params(int ac, char **av);
```

Chapter IX

Exercice 06 : ft_list_clear

	Exercice : 06
ft_list_clear	
Turn-in directory : $ex06/$	
Files to turn in : `ft_list_clear.c, ft_list.h`	
Allowed functions : `free`	
Remarks : `n/a`	

- Create the function `ft_list_clear` which clears all links from the list.
- It'll then assign the list's pointer to null.
- Here's how it should be prototyped :

```
void ft_list_clear(t_list **begin_list);
```

Chapter X

Exercice 07 : ft_list_at

	Exercice : 07
ft_list_at	
Turn-in directory : *ex*07/	
Files to turn in : `ft_list_at.c, ft_list.h`	
Allowed functions : `Nothing`	
Remarks : `n/a`	

- Create the function `ft_list_at` which returns the Nth element of the list.
- 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);
```