B1- C-Pool

B-CPE-042

Day 11

Linked lists

v1.4

Day 11

Linked lists

repository name: : CPool_Day11
repository rights: : ramassage-tek
language: : C
group size: : 1
allowed functions: : write, malloc, free

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- Don't push your **main** function into your delivery directory, we will be adding our own. Your files will be compiled adding our **main.c**.
- If one of your files prevents you from compiling with * **.c**, the Autograder will not be able to correct your work and you will receive a 0.

All **.c** files from your delivery folder will be collected and compiled with your **libmy**, which is found in **CPool_Day11/lib/my**. For those of you using **.h** files, they must be located in **CPool_Day11/include**. The Autograder will compile your library and your includes.

For the tasks regarding lists, we will be using a slightly different structure from the one seen during the lesson:

```
typedef struct          s_list
{
    void                *data;
    struct s_list       *next;
}                       t_list;
```

This structure must be found in a file named, **mylist.h** in your includes folder.

Create your repository at the beginning of the day and submit your work on a regular basis!
The delivery directory is specified within the instructions for each task.
In order to keep your repository clean, pay attention to `gitignore`.

Task 0

Unit Tests

It is highly recommended to test your functions as you develop them. It is common practice to create a function named `main` (and a designated file to host it) to check the functions separately.
Create a directory named `tests`.
Create a `main` function within a file named `tests-$FUNCTION_NAME.c`, to be stored in the `tests` directory named.
This function must contain all the necessary calls to the task function in order to cover all of the function's possible situations (normal or irregular).

Task 1

my_params_in_list

Write a function named **my_params_in_list** that creates a new list from the command line arguments. The address of the list's first node is returned.
It must be prototyped as follows:

```
t_list *my_params_in_list(int ac, char **av);
```

Delivery: CPool_Day11/task01/my_params_in_list.c
For instance,

Terminal + x

```
~/B-CPE-042> ~/B-CPE-042> ./a.out test arg2 arg3
```

If the main function directly transmits its **argc/argv** arguments to my_params_in_list, the function must place **./a.out** first on the list, then **test**, **arg2** and **arg3**.
When scanning the list, we will have **arg3** as the first element, then **arg2**, ... and finally, **./a.out**.

Task 2

my_list_size

Write a function called **my_list_size** that returns the number of elements on the list.
It must be prototyped as follows:

```
int my_list_size(t_list *begin);
```

Delivery: CPool_Day11/task02/my_list_size.c

Task 3

my_rev_list

Write a function named **my_rev_list** that reverses the order of the list's elements.
It should be prototyped as follows:

```
int my_rev_list(t_list **begin);
```

Delivery: CPool_Day11/task03/my_rev_list.c

You are only permitted to manipulate the pointers.

Task 4

my_apply_on_list

Write a function named **my_apply_on_list** that applies a function, given as argument, to the data of each node on the list. It must be prototyped as follows:

```
int my_apply_on_list(t_list *begin, int (*f)(void*));
```

Delivery: CPool_Day11/task04/my_apply_on_list.c

The function pointed by **f** will be used as follows: **(*f)(list_ptr->data);**

Task 5

my_apply_on_eq_in_list

Write a function named **my_apply_on_eq_in_list** that applies a function, given as argument, to the data of certain nodes on the list.

Reference information and a comparison function will enable you to select the proper nodes: **those *equal* to the reference information**

The function must be prototyped as follows:

```
int my_apply_on_eq_in_list(t_list *begin, int (*f) (), void *data_ref, int (*cmp) ());
```

Delivery: CPool_Day11/task05/my_apply_on_eq_in_list.c

The functions pointed by **f** and **cmp** will be used as follows: **(*f)(list_ptr->data);** and **(*cmp)(list_ptr->data, data_ref);**

The **cmp** function could be **my_strcmp**; the elements are only considered equal if *cmp* returns 0 (data is *equal*)

Task 6

my_find_elm_eq_in_list

Write a function named **my_find_elm_eq_in_list** that returns the data of the first node, which is *equal* to the reference data.
It must be prototyped as follows:

```
void *my_find_elm_eq_in_list(t_list *begin, void *data_ref, int (*cmp)());
```

Delivery: CPool_Day11/task06/my_find_elm_eq_in_list.c

Task 7

my_find_node_eq_in_list

Write a function named **my_find_node_eq_in_list** that returns the address of the first node, which contains data *equal* to the reference data.
It must be prototyped as follows:

```
t_list *my_find_node_eq_in_list(t_list *begin, void *data_ref, int (*cmp)());
```

Delivery: CPool_Day11/task07/my_find_node_eq_in_list.c

Task 8

my_rm_all_eq_from_list

Write a function named **my_rm_all_eq_from_list** that erases all elements containing data *equal* to the reference data.
It must be prototyped as follows:

```
int my_rm_all_eq_from_list(t_list **begin, void *data_ref, int (*cmp)());
```

Delivery: CPool_Day11/task08/my_rm_all_eq_from_list.c

Task 9

my_add_list_to_list

Write a function named **my_add_list_to_list** that puts the elements of a **begin2** list at the end of a **begin1** list.
It must be prototyped as follows:

```
int my_add_list_to_list(t_list **begin1, t_list *begin2);
```

Delivery: CPool_Day11/task09/my_add_list_to_list.c

Creating elements is not allowed!

Task 10

my_sort_list

Write a function named **my_sort_list** that sorts a list's contents in ascending order by comparing data, node-to-node, with a comparison function.
It must be prototyped as follows:

```
int my_sort_list(t_list **begin, int (*cmp)());
```

Delivery: CPool_Day11/task10/my_sort_list.c

Task 11

my_put_elem_in_sort_list

Write a function named **my_put_elem_in_sort_list** that creates a new element and insert it into an ordered list, so that the list remains sorted in ascending order.
It must be prototyped as follows:

```
int my_put_elem_in_sort_list(t_list **begin, void *data, int (*cmp)());
```

Delivery: CPool_Day11/task11/my_put_elem_in_sort_list.c

Task 12

my_add_sort_list_to_sort_list

Write a function named **my_add_sort_list_to_sort_list** that integrates the elements of an ordered list, **begin2**, into another ordered list, **begin1**, so that **begin1** remains sorted in ascending order.
It must be prototyped as follows:

```
int my_add_sort_list_to_sort_list(t_list **begin1, t_list *begin2, int (*cmp) ());
```

Delivery: CPool_Day11/task12/my_add_sort_list_to_sort_list.c

Watch out for **NULL** pointers!