B2- EpiReboot

B-EPI-050

Pingouin

Linked Lists

Pingouin

Linked Lists

binary name:	
repository name:	none
repository rights:	ramassage-tek
language:	C
group size:	1
compilation:	gcc or Makefile

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Fonctions autorisées :

- malloc
- free

Task 00

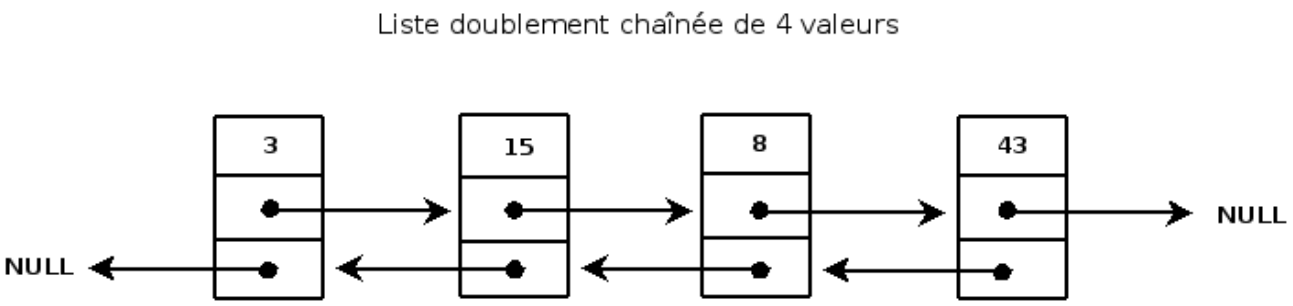
What is a linkedlist ?

Le principe d'une liste chaînée repose sur le principe de "noeuds" qui sont liés.
Ainsi la structure d'un noeud est:

- -Un pointeur vers le noeud suivant.
- -Un pointeur vers le noeud précédent.
- -Une valeur (ici ce sera un integer).

On utilisera également la structure t_linkedlist qui contiendra:

- -Une longueur.
- -Un pointeur vers le noeud de début.
- -Un pointeur vers le noeud de fin.



Utilisez cette structure:

Terminal

```
typedef struct s_node
{
   struct s_node *next;
   struct s_node *prev;
   int value;
}              t_node;

typedef struct s_linkedlist
{
   size_t length;
   t_node *begin;
   t_node *end;
}              t_linkedlist;
```

Votre premier exercice va être d'initialiser le contenu de votre structure.
Voici le prototype de la fonction d'initialisation:

Terminal

```
int linkedlis_init(t_linkedlist *list);
```

Task 01

Get some elements

Pour pouvoir comprendre.

Votre tâche est, si vous l'acceptez, de faire des fonctions permettant d'ajouter un élément au début, au milieu et à la fin de cette liste.

Voici les prototypes:

struct.h :

Terminal

```
/* Retourne le pointeur sur le noeud à l'index indiqué */.
t_node *linkedlist_getnode(t_linkedlist *list, size_t index);

/* Ajout à la fin */
int linkedlist_append(t_linkedlist *list, int element);

/* Ajout au début */
int linkedlist_prepend(t_linkedlist *list, int element);

/* Insert un élément à l'index indiqué */
int linkedlist_insert(t_linkedlist *list, int elememement, size_t index);
```

Découpez bien votre code en plusieurs fonctions. (e.g: une fonction pour ajouter un élément si la liste est vide)

Task 02

Let some elements

Pour pouvoir comprendre.

Votre tâche est, si vous l'acceptez, de faire des fonctions permettant de supprimer un élément au début, au milieu et à la fin de cette liste.

Voici les prototypes:

struct.h :

Terminal

```
/* Supprimer l'élément du début */
int linkedlist_remove_begin(t_linkedlist *list);

/* Supprimer l'élément de fin */
int linkedlist_remove_end(t_linkedlist *list);

/* Supprimer l'élément à l'index indiqué */
int linkedlist_remove_index(t_linkedlist *list, size_t index);
```

Découpez bien votre code en plusieurs fonctions. (e.g: une fonction pour ajouter un élément si la liste est vide)