Tasks

0. Print list

**mandatory**

Write a function that prints all the elements of a list\_t list.

* Prototype: size\_t print\_list(const list\_t \*h);
* Return: the number of nodes
* Format: see example
* If str is NULL, print [0] (nil)
* You are allowed to use printf

julien@ubuntu:~/Singly linked lists$ cat 0-main.c

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include "lists.h"

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

list\_t \*head;

list\_t \*new;

list\_t hello = {"World", 5, NULL};

size\_t n;

head = &hello;

new = malloc(sizeof(list\_t));

if (new == NULL)

{

printf("Error\n");

return (1);

}

new->str = strdup("Hello");

new->len = 5;

new->next = head;

head = new;

n = print\_list(head);

printf("-> %lu elements\n", n);

printf("\n");

free(new->str);

new->str = NULL;

n = print\_list(head);

printf("-> %lu elements\n", n);

free(new);

return (0);

}

julien@ubuntu:~/Singly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main.c 0-print\_list.c -o a

julien@ubuntu:~/Singly linked lists$ ./a

[5] Hello

[5] World

-> 2 elements

[0] (nil)

[5] World

-> 2 elements

julien@ubuntu:~/Singly linked lists$

**Repo:**

* GitHub repository: holbertonschool-low\_level\_programming
* Directory: singly\_linked\_lists
* File: 0-print\_list.c

Review your work Get a sandbox

**9/9**pts

1. List length

**mandatory**

Write a function that returns the number of elements in a linked list\_t list.

* Prototype: size\_t list\_len(const list\_t \*h);

julien@ubuntu:~/Singly linked lists$ cat 1-main.c

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include "lists.h"

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

list\_t \*head;

list\_t \*new;

list\_t hello = {"World", 5, NULL};

size\_t n;

head = &hello;

new = malloc(sizeof(list\_t));

if (new == NULL)

{

printf("Error\n");

return (1);

}

new->str = strdup("Hello");

new->len = 5;

new->next = head;

head = new;

n = list\_len(head);

printf("-> %lu elements\n", n);

free(new->str);

free(new);

return (0);

}

julien@ubuntu:~/Singly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main.c 1-list\_len.c -o b

julien@ubuntu:~/Singly linked lists$ ./b

-> 2 elements

julien@ubuntu:~/Singly linked lists$

**Repo:**

* GitHub repository: holbertonschool-low\_level\_programming
* Directory: singly\_linked\_lists
* File: 1-list\_len.c

Review your work Get a sandbox

**8/8**pts

2. Add node

**mandatory**

Write a function that adds a new node at the beginning of a list\_t list.

* Prototype: list\_t \*add\_node(list\_t \*\*head, const char \*str);
* Return: the address of the new element, or NULL if it failed
* str needs to be duplicated
* You are allowed to use strdup

julien@ubuntu:~/Singly linked lists$ cat 2-main.c

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include "lists.h"

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

list\_t \*head;

head = NULL;

add\_node(&head, "Alexandro");

add\_node(&head, "Asaia");

add\_node(&head, "Augustin");

add\_node(&head, "Bennett");

add\_node(&head, "Bilal");

add\_node(&head, "Chandler");

add\_node(&head, "Damian");

add\_node(&head, "Daniel");

add\_node(&head, "Dora");

add\_node(&head, "Electra");

add\_node(&head, "Gloria");

add\_node(&head, "Joe");

add\_node(&head, "John");

add\_node(&head, "John");

add\_node(&head, "Josquin");

add\_node(&head, "Kris");

add\_node(&head, "Marine");

add\_node(&head, "Mason");

add\_node(&head, "Praylin");

add\_node(&head, "Rick");

add\_node(&head, "Rick");

add\_node(&head, "Rona");

add\_node(&head, "Siphan");

add\_node(&head, "Sravanthi");

add\_node(&head, "Steven");

add\_node(&head, "Tasneem");

add\_node(&head, "William");

add\_node(&head, "Zee");

print\_list(head);

return (0);

}

julien@ubuntu:~/Singly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main.c 2-add\_node.c 0-print\_list.c -o c

julien@ubuntu:~/Singly linked lists$ ./c

[3] Zee

[7] William

[7] Tasneem

[6] Steven

[9] Sravanthi

[6] Siphan

[4] Rona

[4] Rick

[4] Rick

[7] Praylin

[5] Mason

[6] Marine

[4] Kris

[7] Josquin

[4] John

[4] John

[3] Joe

[6] Gloria

[7] Electra

[4] Dora

[6] Daniel

[6] Damian

[8] Chandler

[5] Bilal

[7] Bennett

[8] Augustin

[5] Asaia

[9] Alexandro

julien@ubuntu:~/Singly linked lists$

**Repo:**

* GitHub repository: holbertonschool-low\_level\_programming
* Directory: singly\_linked\_lists
* File: 2-add\_node.c

Review your work Get a sandbox

**0/9**pts

3. Add node at the end

**mandatory**

Write a function that adds a new node at the end of a list\_t list.

* Prototype: list\_t \*add\_node\_end(list\_t \*\*head, const char \*str);
* Return: the address of the new element, or NULL if it failed
* str needs to be duplicated
* You are allowed to use strdup

julien@ubuntu:~/Singly linked lists$ cat 3-main.c

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include "lists.h"

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

list\_t \*head;

head = NULL;

add\_node\_end(&head, "Anne");

add\_node\_end(&head, "Colton");

add\_node\_end(&head, "Corbin");

add\_node\_end(&head, "Daniel");

add\_node\_end(&head, "Danton");

add\_node\_end(&head, "David");

add\_node\_end(&head, "Gary");

add\_node\_end(&head, "Holden");

add\_node\_end(&head, "Ian");

add\_node\_end(&head, "Ian");

add\_node\_end(&head, "Jay");

add\_node\_end(&head, "Jennie");

add\_node\_end(&head, "Jimmy");

add\_node\_end(&head, "Justin");

add\_node\_end(&head, "Kalson");

add\_node\_end(&head, "Kina");

add\_node\_end(&head, "Matthew");

add\_node\_end(&head, "Max");

add\_node\_end(&head, "Michael");

add\_node\_end(&head, "Ntuj");

add\_node\_end(&head, "Philip");

add\_node\_end(&head, "Richard");

add\_node\_end(&head, "Samantha");

add\_node\_end(&head, "Stuart");

add\_node\_end(&head, "Swati");

add\_node\_end(&head, "Timothy");

add\_node\_end(&head, "Victor");

add\_node\_end(&head, "Walton");

print\_list(head);

return (0);

}

julien@ubuntu:~/Singly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-main.c 3-add\_node\_end.c 0-print\_list.c -o d

julien@ubuntu:~/Singly linked lists$ ./d

[4] Anne

[6] Colton

[6] Corbin

[6] Daniel

[6] Danton

[5] David

[4] Gary

[6] Holden

[3] Ian

[3] Ian

[3] Jay

[6] Jennie

[5] Jimmy

[6] Justin

[6] Kalson

[4] Kina

[7] Matthew

[3] Max

[7] Michael

[4] Ntuj

[6] Philip

[7] Richard

[8] Samantha

[6] Stuart

[5] Swati

[7] Timothy

[6] Victor

[6] Walton

julien@ubuntu:~/Singly linked lists$

**Repo:**

* GitHub repository: holbertonschool-low\_level\_programming
* Directory: singly\_linked\_lists
* File: 3-add\_node\_end.c

Review your work Get a sandbox

**0/9**pts

4. Free list

**mandatory**

Write a function that frees a list\_t list.

* Prototype: void free\_list(list\_t \*head);

julien@ubuntu:~/Singly linked lists$ cat 4-main.c

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include "lists.h"

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

list\_t \*head;

head = NULL;

add\_node\_end(&head, "Bob");

add\_node\_end(&head, "&");

add\_node\_end(&head, "Kris");

add\_node\_end(&head, "love");

add\_node\_end(&head, "asm");

print\_list(head);

free\_list(head);

head = NULL;

return (0);

}

julien@ubuntu:~/Singly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-main.c 4-free\_list.c 3-add\_node\_end.c 0-print\_list.c -o e

julien@ubuntu:~/Singly linked lists$ valgrind ./e

==3598== Memcheck, a memory error detector

==3598== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.

==3598== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info

==3598== Command: ./e

==3598==

[6] Bob

[1] &

[3] Kris

[4] love

[3] asm

==3598==

==3598== HEAP SUMMARY:

==3598== in use at exit: 0 bytes in 0 blocks

==3598== total heap usage: 11 allocs, 11 frees, 1,166 bytes allocated

==3598==

==3598== All heap blocks were freed -- no leaks are possible

==3598==

==3598== For counts of detected and suppressed errors, rerun with: -v

==3598== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

julien@ubuntu:~/Singly linked lists$

**Repo:**

* GitHub repository: holbertonschool-low\_level\_programming
* Directory: singly\_linked\_lists
* File: 4-free\_list.c