Exercises — Stack

version **#7be580532266ed398481e31366afcc24b1950c2a**

Assistants C/Unix 2023 <assistants@tickets.assistants.epita.fr>

Copyright

This document is for internal use at EPITA (website) only.

Contents

*https://intra.assistants.epita.fr

File Tree

```
stack/
├── stack.c  (to submit)
└── stack.h
```

Authorized functions : You are only allowed to use the following functions

- malloc(3)
- calloc(3)
- free(3)

Authorized headers : You are only allowed to use the functions defined in the following headers

- err.h
- errno.h
- assert.h
- stddef.h

Compilation : Your code must compile with the following flags

- -std=c99 -pedantic -Werror -Wall -Wextra -Wvla

Main function : None

1 Goal

You have to implement a stack (a LIFO data structure) of integers.

The structure used for this exercise is the following:

```c
struct stack
{
    int data;
    struct stack *next;
};
```

1.1 Insertion

```c
struct stack *stack_push(struct stack *s, int e);
```

This function adds the `e` element on top of the stack `s` and returns the top of the stack after the insertion. If any error occurs, you have to return `s` as it was when your function was called.

Be careful!

Calling `stack_push(NULL, 42)` is not an error, it should return a stack with a single element, 42.

1.2 Remove

```
struct stack *stack_pop(struct stack *s);
```

This function removes the top of the stack `s` and returns the new top. If `s` is `NULL`, returns `NULL`.

1.3 Get Value

```
int stack_peek(struct stack *s);
```

This function returns the data stored at the top of the stack. You do not have to handle the case where `s` is `NULL`.

The way is lit. The path is clear. We require only the strength to follow it.