EXERCISES — Binary Search With Pointers

version **#7be580532266ed398481e31366afcc24b1950c2a**

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

ASSISTANTS C/UNIX 2023 <assistants@tickets.assistants.epita.fr>

Contents

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

File Tree

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

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 Binary search

When looking for an element in a sorted array, it is possible to get the result with a logarithmic complexity (which means fast) using *dichotomy*. Here is a quick reminder of how it works:

```
Search of 42 inside: [ 0 1 4 5 9 10 18 22 42 51 69 ]

[ 0 1 4 5 9 10 18 22 42 51 69 ]
<           ^                 >

[ 0 1 4 5 9 10 18 22 42 51 69 ]
               <     ^        >
```

A header (`bsearch.h`) containing all the required functions is provided on the intranet.

You **must** add this header to your submission, and you **must not** modify it.

2 Goal

```c
int *binary_search(int *begin, int *end, int elt);
```

- `begin` and `end` will never be `NULL`.
- Search `elt` in the memory range of `[begin, end[`.
  - `begin` is a pointer to the first element.
  - `end` is a pointer **after** the last element.
- All the elements in the memory range of `[begin, end[` are guaranteed to be sorted in ascending order.

- The array does not contain any duplicate elements.
- An empty range is represented by `begin == end`.
- If `elt` is found, return a pointer to the element.
- If `elt` is not found, return a pointer to the memory location where `elt` should be inserted to keep the array sorted.
- If the array is empty, return `begin`.

3 Examples

```c
int main(void)
{
    int a[] = { 0, 1, 4, 5, 9, 10, 18, 22, 42, 51, 69 };

    assert(binary_search(a, a + 11, 5) == a + 3);

    assert(binary_search(a, a + 11, 0) == a);

    assert(binary_search(a, a + 11, -1) == a);

    assert(binary_search(a, a + 11, 99) == a + 11);

    assert(binary_search(a, a + 11, 68) == a + 10);

    assert(binary_search(a, a, 5) == a);
}
```

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