EXERCISES — Binary search with poir

version #

ASSISTANTS C/UNIX 2022 <assistants@tickets.assistants.epita.fr>

Copyright

The use of this document must abide by the following rules:

- ▷ You downloaded it from the assistants' intranet.*
- ▷ This document is strictly personal and must **not** be passed onto someone else.
- ▷ Non-compliance with these rules can lead to severe sanctions.

Contents

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

1 Binary search with pointers

Files to submit:

- binary_search_ptr/bsearch.c
- binary_search_ptr/bsearch.h

Provided files:

- binary_search_ptr/bsearch.h

Main function: None

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

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

1.1 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.

1.2 Goal

```
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`.

1.3 Examples

```
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);
```

It is my job to make sure you do yours.