Lab 9 - Exercise – All sections
qsort()
CS 262 – Spring 2022

The purpose of this activity is to practice how to use the `qsort()` function to sort values in an array. This function implements the **Quick Sort Algorithm** for sorting.

Background:

The standard C library `<stdlib.h>` provides the `qsort()` function that can be used for sorting an array. The prototype of `qsort()` is:

```
  void qsort(void* base, size_t num, size_t size,
                            int(*comp)(const void*, const void*));
```

Parameters:

`base`	Pointer to the first element of the array to be sorted.
`num`	Number of elements in the array pointed by base.
`size`	It's the size in bytes of each element in the array
`comp`	Function that compares two elements.

Return Value

`void`	It does not return any value

The `comp` Function

```
  int comp (const void* p1, const void* p2)
```

The comparator function takes two pointers as parameters and returns an integer (negative, zero or positive) which indicates the result of the comparison of the elements pointed by the pointers.

`<0`	The element pointed by `p1` goes before the element pointed by `p2`
`0`	The element pointed by `p1` is equivalent to the element pointed by `p2`
`>0`	The element pointed by `p1` goes after the element pointed by `p2`

Example:

How to implement `comp` to compare two `int` elements to sort them in ascending:

```
int comp(const void* p1, const void* p2){
    int element1 = *(const int*)p1;
    int element2 = *(const int*)p2;

    if(element1 < element2) return -1;
    if(element1 > element2) return 1;
    return 0;
}
```

Description of the program

Code a program that prompts the user to enter an input number (`N`)

Use `malloc()` or `calloc()` to create an array of `N` numbers, the elements will be `float`

- In a loop ask the user to input each of the elements and store them in the array
- Print the content of the array
- Call the `qsort()` function to sort the elements in descending order
- Print the content of the sorted array

Requirements:

- Use `fgets()` and `sscanf()` to get the user input.
- Use `calloc()` or `malloc()` to ask for *dynamic memory* for the array
- Free the *dynamic memory* at the end of the program
- Prints each `float` number rounded to one decimal

Example 1:

```
Enter a number: 3
Enter the element 1: 5.8
Enter the element 2: 2.3
Enter the element 3: 4.5

Original array:
5.8  2.3  4.5

Sorted array:
5.8  4.5  2.3
```

Example 2:

```
Enter a number: 4
Enter the element 1: 1
Enter the element 2: 8.2
Enter the element 3: -3.6
Enter the element 4: 100.5

Original array:
1.0  8.2  -3.6 100.5

Sorted array:
100.5 8.2 1.0  -3.6
```