Part A: Search algorithms

1. Write a C program to find the position of a target value within a sorted array using binary search. You can assume that you have a static array initialized with sorted values.
2. Write a C program to find the position of a target value within an array using linear search. You can assume that you have a static array initialized with sorted values.
3. Write a program to solve the following problem:
    a. Read 10 integers from the standard input and store them in an array.
    b. Find all negative values and replace them with their absolute value.
    c. Display the number of negative values entered by the user.

Part B: Sort algorithms

1. Write a C program to sort a list of integer elements using the insertion-sort algorithm. You can assume that you have a static array initialized with values or you can use random function `rand()` to initialize the values to be sorted.
2. Write a C program to sort a list of integer elements using the bubble sort algorithm. You can assume that you have a static array initialized with values or you can use random function `rand()` to initialize the values to be sorted.
3. Merge Sort
    a. Analyze the algorithm. You can look at the web page https://www.geeksforgeeks.org/merge-sort/
    b. Analyze the code, try to understand it but take into consideration that this algorithm uses recursion (recursive functions) that we haven't seen yet.
    c. Check the result (at the end of each phase) for the following array (12,2,16,30,8,28,4,10,20,6,18)
4. Quick Sort
    a. Analyze the algorithm. You can look at the web page https://www.geeksforgeeks.org/quick-sort/
    b. Analyze the code, try to understand it but take into consideration that this algorithm uses recursion (recursive functions) that we haven't seen yet.
    c. Check the result (at the end of each phase) for the following array (12,2,16,30,8,28,4,10,20,6,18)
    d. Show that this algorithm takes O(n^2) time when the input list is already sorted.
5. Compare times for all the sorting algorithms you have implemented.

Notes:

- You can use random function `rand()` to initialize the values to be sorted.
- For analyzing the execution time of different configurations of the program, it can take different number of elements to sort, for example 10, 100, 1000, 10000.
- For calculating the execution time, you can use the function `gettimeofday()` that returns the time expressed in elapsed seconds and microseconds since 00:00:00, January 1, 1970 (Unix Epoch). You will have to include the library <sys/time.h>.

  Here you have an example on how to use this function:

  ```
  struct timeval start, end;
  gettimeofday(&start, NULL);
  /* here, do your time-consuming job */
  gettimeofday(&end, NULL);
  printf("Execution time is: %ld micro seconds\n",
      ((end.tv_sec * 1000000 + end.tv_usec) -
      (start.tv_sec * 1000000 + start.tv_usec)));
  ```

6. Convert an array [10, 26, 52, 76, 13, 8, 3, 33, 60, 42] into a balanced binary search tree. Give an example figure of this tree.
7. Write a C program to sort names in alphabetical order. You can assume that you have a static array initialized with names.