Binary search implementation

Things to ask and consider

- Is the array already sorted or do I have to sort it?
- How big is the input array

Initial algorithm

- check if the start index is less than to the end index
    - if start index is less than the end index
        - find middle index
        - compare middle index with search value
            - if middle index is equal to search value
                - return middle index
            - if middle index is greater than the search value
                - change the end index to middle index and recursive call
            - if middle index is less than the search value
                - change the start index to middle index and recursive call
    - if start index is greater than the end index
        - return -1

C++ Code:

```
#include <iostream>

int binarySearch(int list[], int start, int end, int value){
	//check base case
	if(start <= end){
		int mid_index = (start + end) / 2;
		int mid_val = list[mid_index];

		// check if  both are the same
		if(mid_val == value)
			return mid_index;
		else if(mid_val > value)
			return binarySearch(list,start, mid_index - 1, value);
		else if(mid_val < value)
			return binarySearch(list, mid_index + 1, end, value);
	}
	else{
		return -1;
	}
}

//driver code
int main(){
	int list[10] = {10,20,30,50,60,80,110,130,140,170};
	int len_list = sizeof(list)/sizeof(*list);
	int value = 130;
	int index = binarySearch(list,0, len_list,value);
	std::cout << “The value “ << value << “is at index: “ << index <<std::endl;

	value = 200;
	 index = binarySearch(list,0, len_list,value);
	std::cout << "The value " << value << " is at index: " << index <<std::endl;

	value = -100;
	index = binarySearch(list,0, len_list,value);
	std::cout << "The value " << value << " is at index: " << index <<std::endl;

}
```