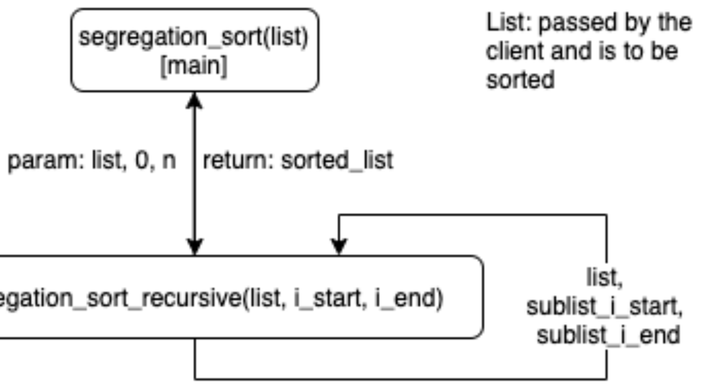
Lab 10: Segregation Sort Design

Structure Chart



```
FUNCTION segregation sort recursive

param: list (passed by reference)
param: i_start
param: i_end

return: sorted_list

    // recursive end-condition
    IF i_start == i_end OR size of list == 0:
        RETURN list
        // this either means the sub-list is sorted
        // or the entire list is sorted.
        // In either case, we return the list
        // to indicate the task/subtask is done.

    // else

    // define
    i_up    = i_start // upward bound counter, starts at the beginning and moves up
    i_down  = i_end   // lower bound counter, starts at the end and moves down
    i_pivot = average_value_of(i_start, i_end)
    pivot_val = list[i_pivot]
```

```
// find value in lower part of list that is greater than pivot
FOR i in range(i_start, i_pivot):

    // update swap index...
    i_up = i

    // is lower val greater than center val?
    IF list[i] > pivot_val:
        // lower val is out of order, we've found
        // the value we want to swap
        // exit the loop to mark it for swap
        BREAK
        // (if all values are in order,
        // i_pivot is marked and nothing happens)

// same as the lower:
// find value in lower part of list that is greater than pivot
FOR i in range(i_end, i_pivot): // note that this will go backwards

    // update swap index...
    i_down = i

    // is lower val greater than center val?
    IF list[i] > pivot_val:
        // lower val is out of order, we've found
        // the value we want to swap
        // exit the loop to mark it for swap
        BREAK
        // (if all values are in order,
        // i_pivot is marked and nothing happens)

SWAP values at i_up and i_down
```

```
// handle pivot index if it was swapped
// (it was marked as i_up or i_down, swap i_index too)
IF i_up   == i_pivot: i_pivot = i_down
IF i_down == i_pivot: i_pivot = i_up

// recursive part: have function call itself again (recursion)
// to sort upper/lower parts of the list

// sort lower part
segregation_sort_recursive(list, i_up, i_pivot - 1)

// sort upper part
segregation_sort_recursive(list, i_pivot + 1, i_down)

// by this point, the section of the list we want sorted
// should be sorted now, we can return the list as is to
// let the callers (previous variant of this function)
// handle the sorting on the higher level
// when the highest level is taken care of, the list is
// returned sorted
RETURN list
```