Problem: Sorting a list in ascending order with only any amount of right shift of elements

Algorithm: Hill Climbing (Steepest ascent)

```
#Initialize():
    initialize a list -> [7, 1, 9, 0, 5, 8, 4, 2, 10, 0, 20] and return it

#calculate_cost(state):
    Counting Inversion Problem
    for each element of the list:
        look forward in the list and see how many elements are smaller than this element i.e. how
        many are in wrong order
    Add up the number of disorders and return

#generate_neighbors(current_state):
    list = current_state
    neighbors = an empty list
    for each element in the list:
        swap with the forward elements of the list with this element one by one and generate one
        list for each swap using a for loop.
            new_list = newly generated state by shifting the element right n times
            neighbors.append(new_list)
    return neighbors

#State_generation(current_state):
    while True:
        current_state_cost = calculate_cost(current_state)
        print(current_state, current_state_cost )
        min_next_cost = INF
        min_next_state = None
        for each neighbor in generate_neighbors(current_state):
            next_state = neighbor
            next_state_cost = calculate_cost(next_state)

            if next_state_cost is smaller than min_next_cost:
                min_next_cost = next_state_cost
                min_next_state = next_state

        # take that state which has the smallest cost
        if min_next_cost is smaller than current_state_cost:
            current_state = min_next_state
        else :
            print("Final State:", current_state, current_state_cost )
            break


#main():
    state = Initialize()
    State_generation(state)
    FINISH
```

Problem: Sorting a list in ascending order with only any amount of right shift of elements

Algorithm: Hill Climbing (First Choice)

```
#Initialize():
    initialize a list -> [7, 1, 9, 0, 5, 8, 4, 2, 10, 0, 20] and return it

#calculate_cost(state):
    Counting Inversion Problem
    for each element of the list:
        look forward in the list and see how many elements are smaller than this element i.e. how
        many are in wrong order
    Add up the number of disorders and return

#generate_neighbors(current_state):
    list = current_state
    neighbors = an empty list
    for each element in the list:
        swap with the forward elements of the list with this element one by one and generate one
        list for each swap using a for loop.
            new_list = newly generated state by shifting the element right n times
            neighbors.append(new_list)
    return neighbors

#State_generation(current_state):
    while True:
        current_state_cost = calculate_cost(current_state)
        print(current_state, current_state_cost )
        min_next_cost = INF
        min_next_state = None
        for each neighbor in generate_neighbors(current_state):
            next_state = neighbor
            next_state_cost = calculate_cost(next_state)

            if next_state_cost is smaller than current_state_cost :
                min_next_cost = next_state_cost
                min_next_state = next_state
                break

        # take that state which has the smallest cost
        if min_next_cost is smaller than current_state_cost:
            current_state = min_next_state
        else :
            print("Final State:", current_state, current_state_cost )
            break


#main():
    state = Initialize()
    State_generation(state)
    FINISH
```