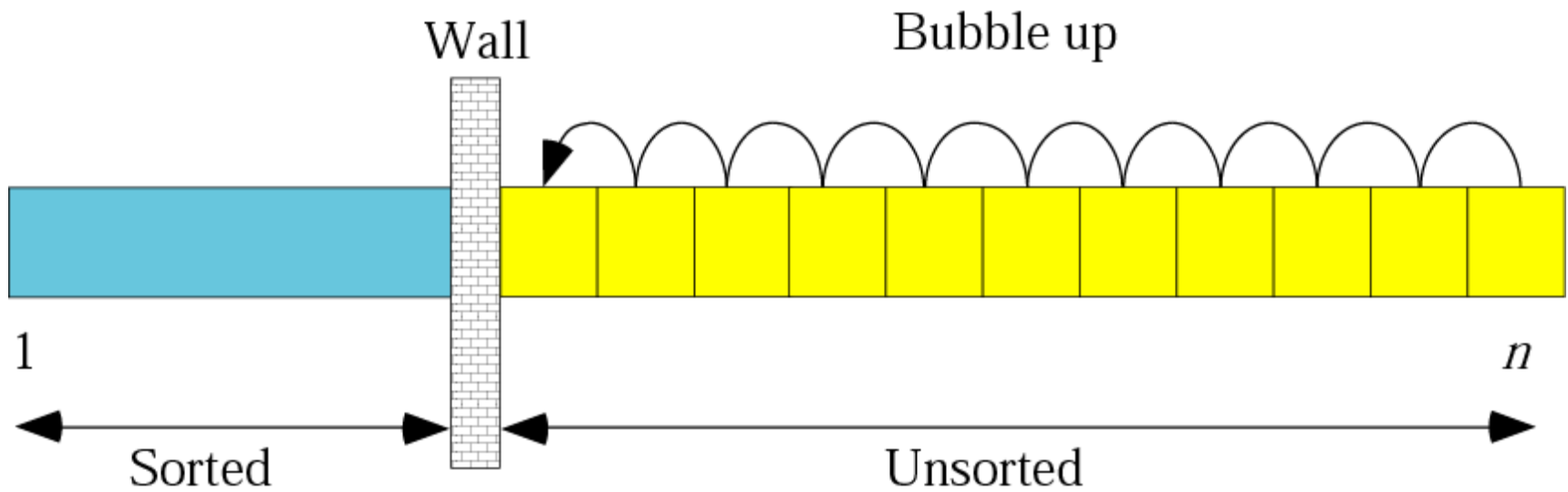# Lab 4

Guideline

Sort with Assembly Language
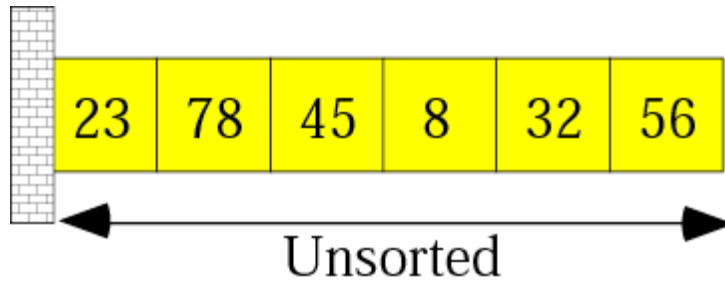
Sorting Algorithm

- Bubble Sort
- Selection Sort
- Insertion Sort
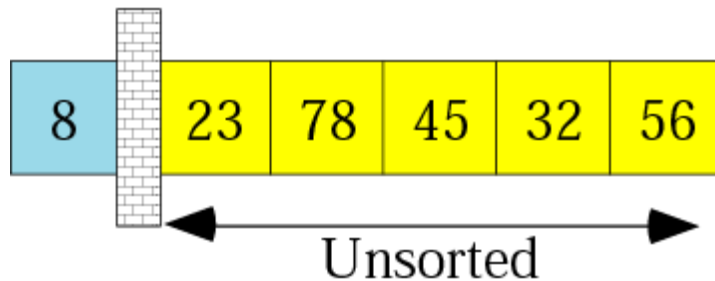- Quick Sort

Bubble Sort

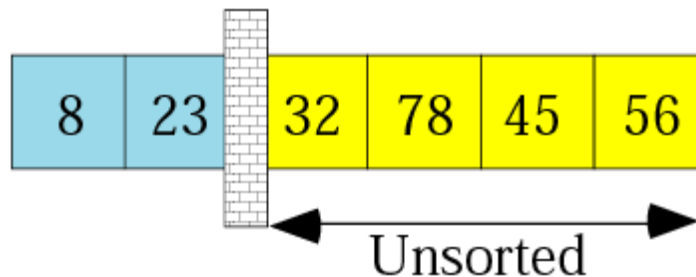# Example of Bubble Sort

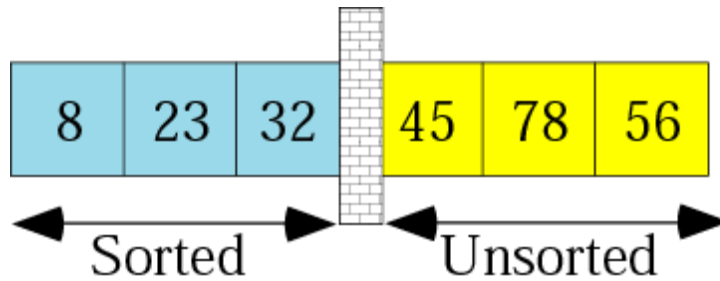| 23 | 78 | 45 | 8 | 32 | 56 |

Unsorted

Original list

| 8 | | 23 | 78 | 45 | 32 | 56 |

Unsorted

After pass 1

| 8 | 23 | | 32 | 78 | 45 | 56 |

Unsorted

After pass 2

Example of Bubble Sort (con)

| 8 | 23 | 32 | | 45 | 78 | 56 |

Sorted — Unsorted

After pass 3

| 8 | 23 | 32 | 45 | | 56 | 78 |
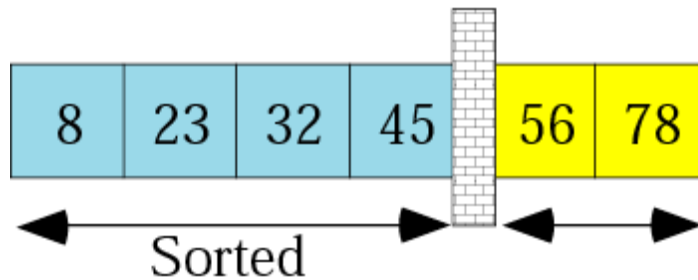
Sorted

After pass 4
Sorted

Selection Sort

Example of Selection Sort

Example of Selection Sort (con)

Insertion Sort

Example of Insertion Sort

| 23 | 78 | 45 | 8 | 32 | 56 |

Unsorted

Original list

| 23 | 78 | 45 | 8 | 32 | 56 |

Unsorted

After pass 1
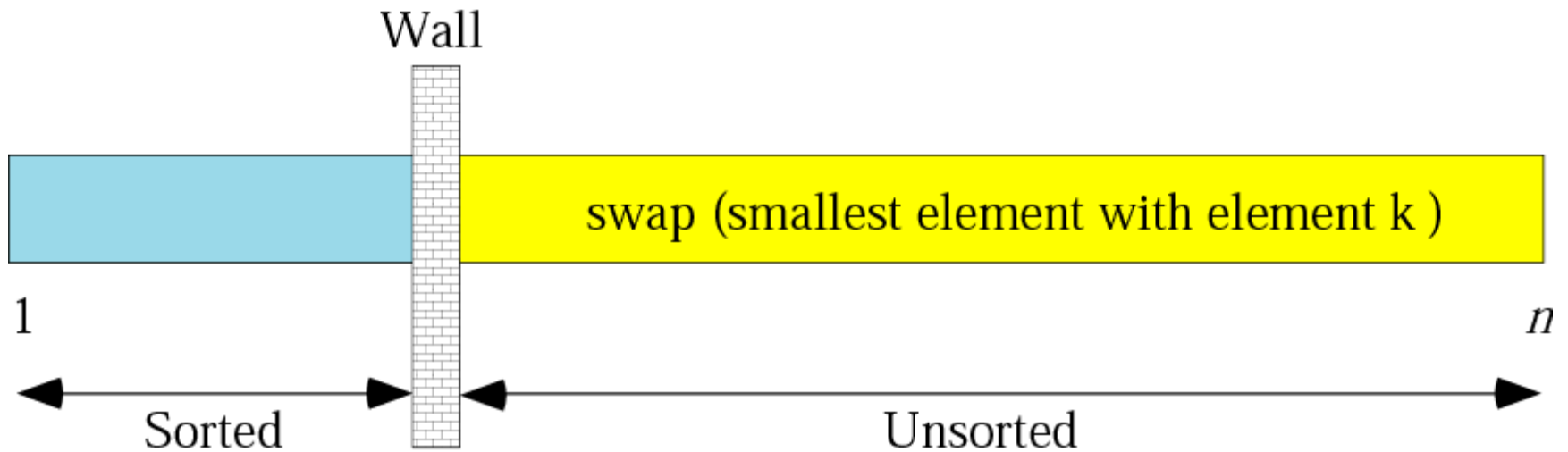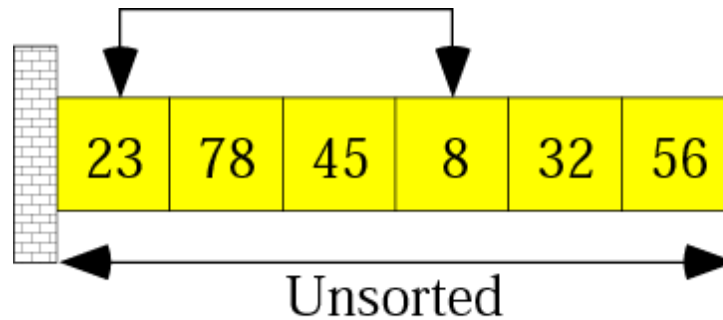
| 23 | 45 | 78 | 8 | 32 | 56 |

Sorted   Unsorted
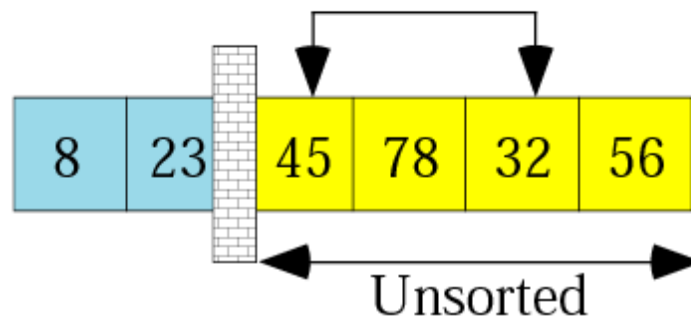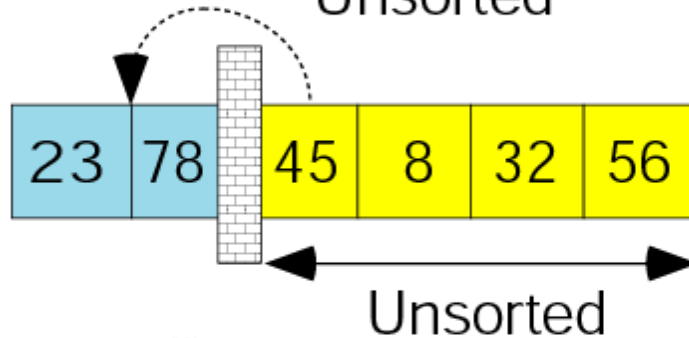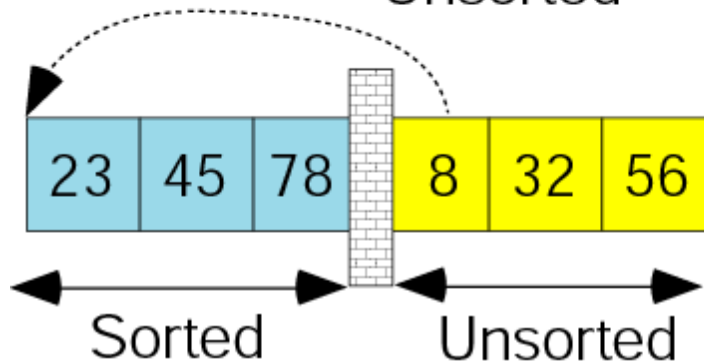
After pass 2

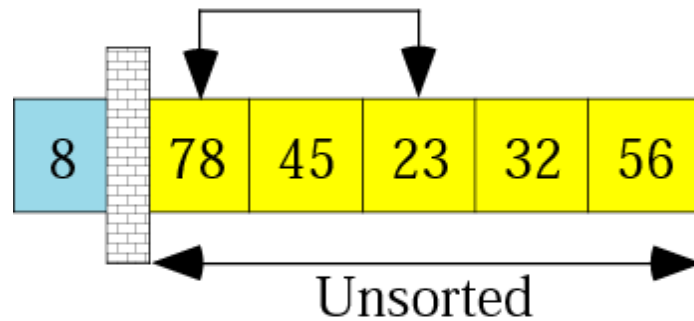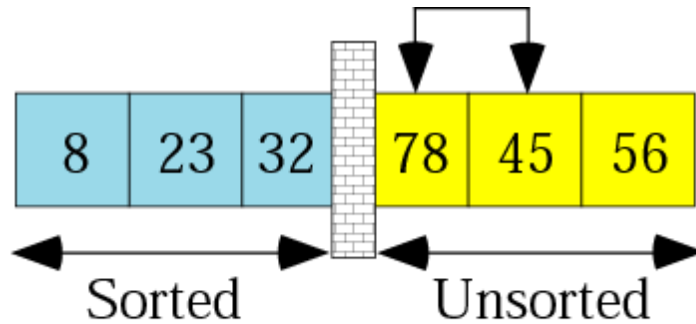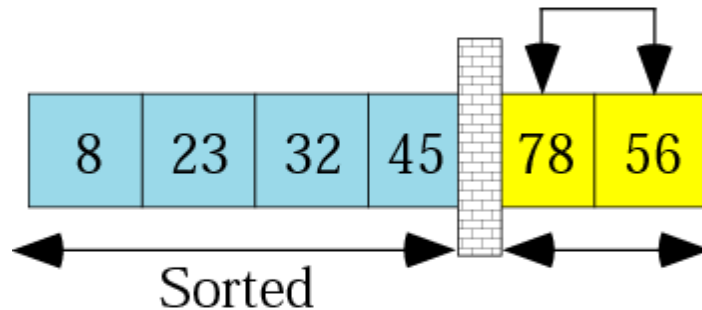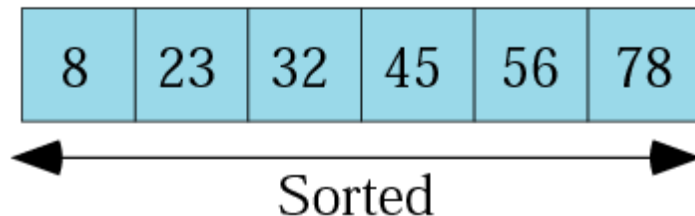# Example of Insertion Sort (con)
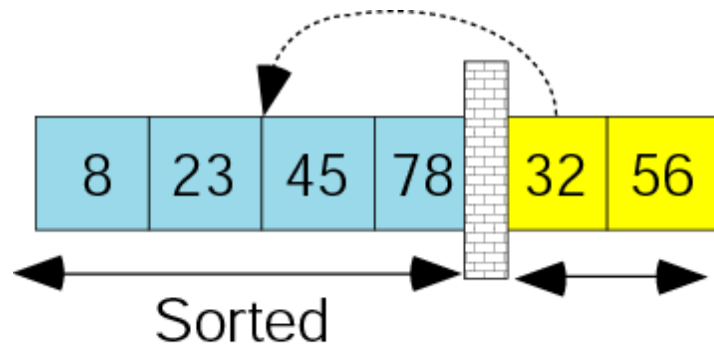
After pass 3

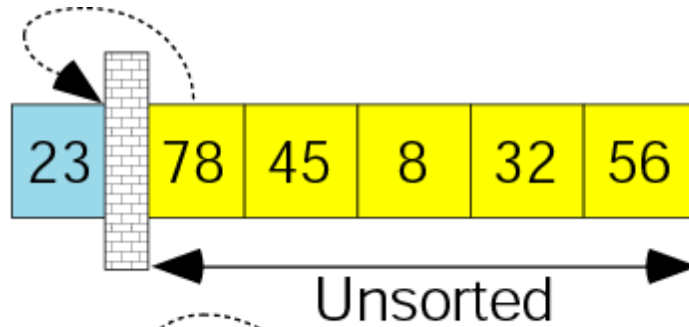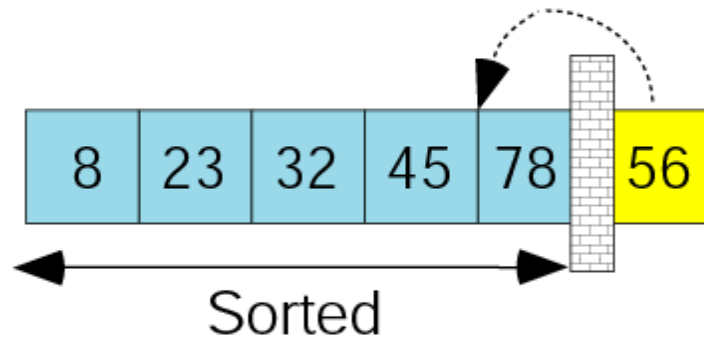After pass 4
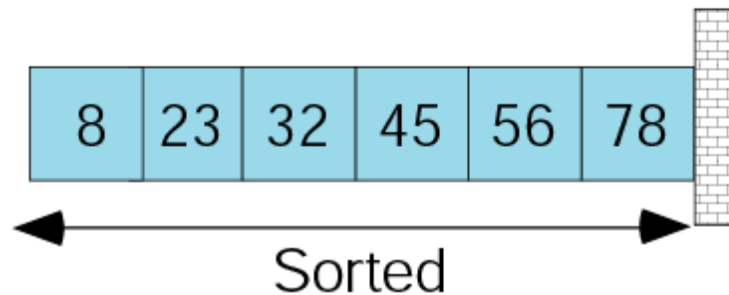
After pass 5

Quick Sort

- Quick Sort is the fastest known sorting algorithm in practice.
- The algorithm is based on divide-and-conquer recursive method.
- The general idea is to
  - pick an element, called a pivot.
  - divide the array of elements into two halves excluding the pivot, v.
  - the first half contains all the elements with the value less than v,
  - the second half has all the elements with the values greater than v.
  - the two halves are then quicksort again recursively.

The Quick Sort Idea

17	14	65	4	22	63	11

Unordered list of values

The Quick Sort Idea

17	14	65	4	22	63	11

Choose pivot value

The Quick Sort Idea

14	4	11	17	65	22	63

Low list
(≤ 17)

High list
(> 17)

The Quick Sort Idea

4	11	14	17	65	22	63

Recursively
apply quicksort
to low list

The Quick Sort Idea

4	11	14	17	22	63	65

Recursively
apply quicksort
to high list

The Quick Sort Idea

4	11	14	17	22	63	65

Sorted list of values

The Quick Sort algorithm

- Pick a pivot, v, and swap it with the last element in an array.
- Start a pointer, i, at position 0 and start a pointer, j, at position $N-1$.
- Move pointer i to the right until i points at an element larger than or equal to v.
- Move pointer j to the left until j points at an element smaller than or equal to v.
- If $i > j$, swap an element that is pointed by i with v.
- If $i < j$, swap an element that is pointed by i with an element that is pointed by j.