Arrays and Pointers

Lecture Objectives:

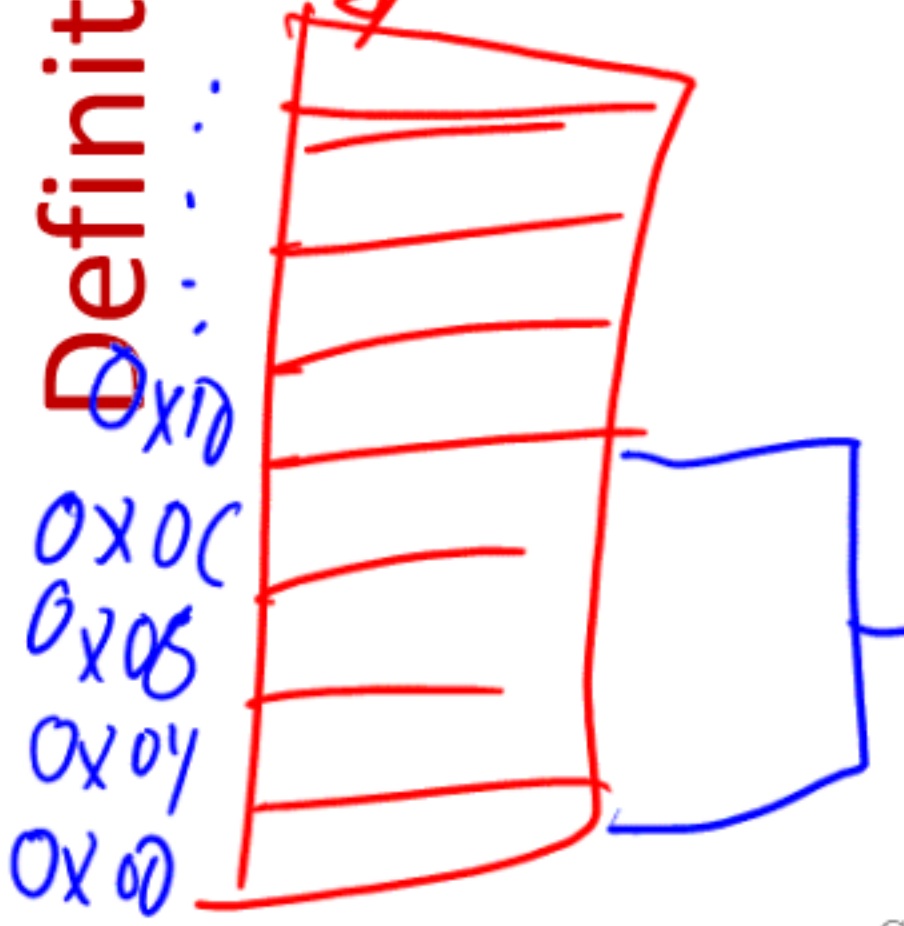
1) Explain how arrays are declared in C.

   Hint: We'll learn a bit about C in doing this.

2) Explain the relationship between arrays and pointers.

- Array
  - A contiguous allocation of memory assigned to a single variable name
- Pointer
  - A variable which holds the address of another variable.

Definitions:



type

Variable name

Size

```
char bob[16];
```

C declaration for an array of 16 bytes / 16 characters.

Definitions

- Array
  - A contiguous allocation of memory assigned to a single variable name
- Pointer
  - A variable which holds the address of another variable.

essentially a reference

Indicates a pointer

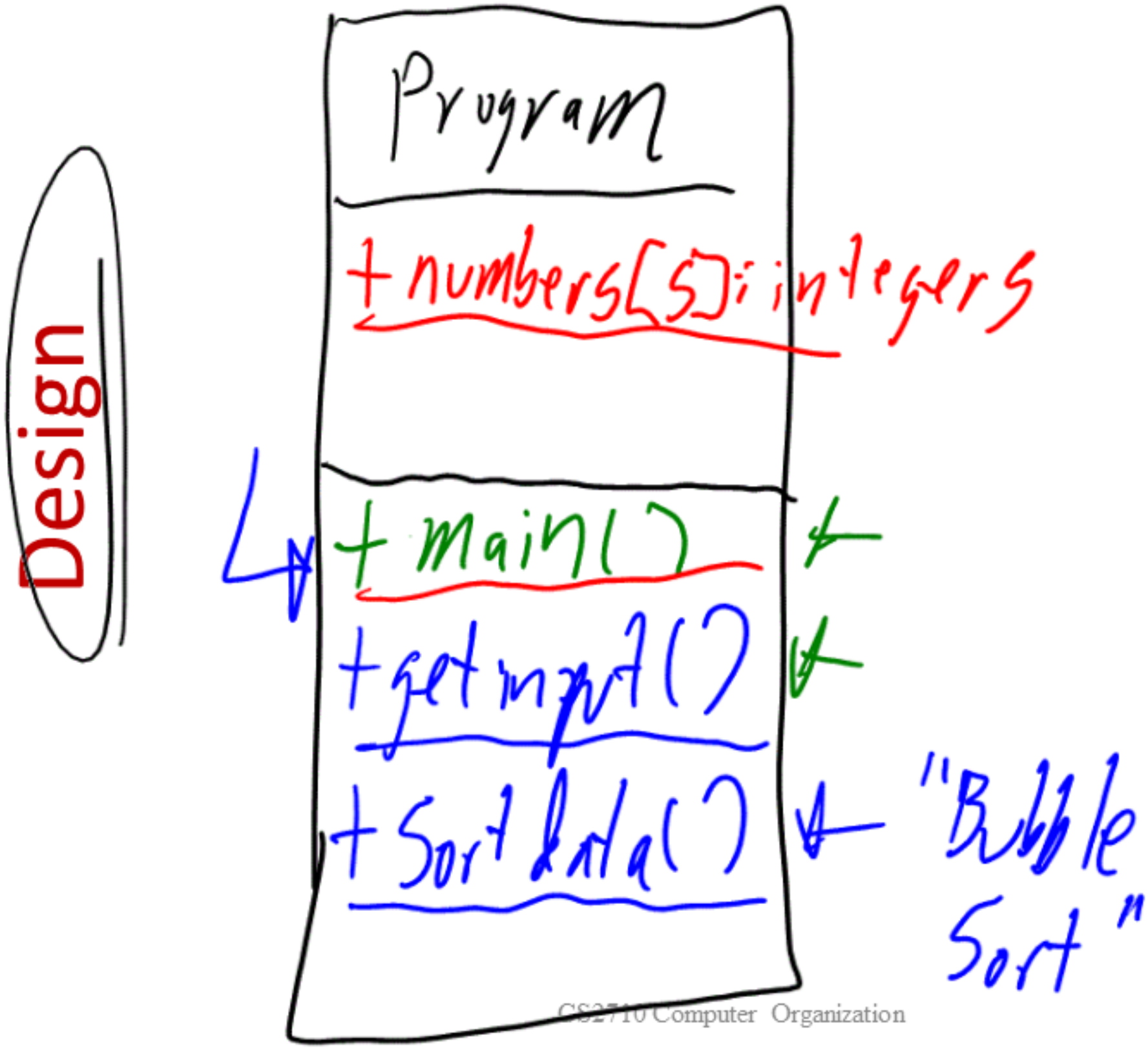
```
char bob[16];
char *bobptr;
```

bobptr: point at the first address of the array

Problem:

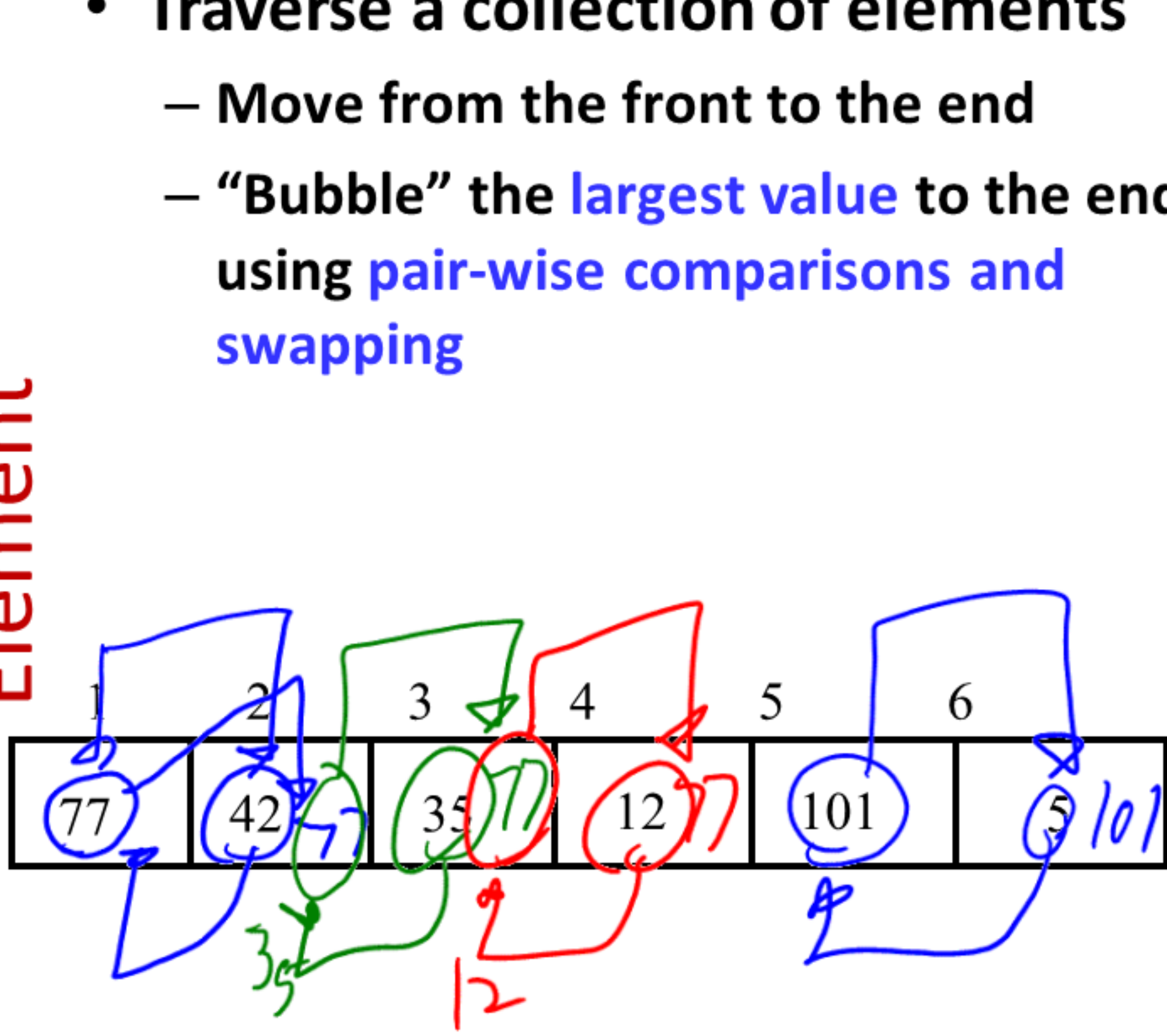
- I want to build a program which will do the following:
  - Prompt the user to enter 5 numbers and store them in an array
  - Sort the 5 numbers in ascending order
  - Print out the sorted number list

- What methods should we write?



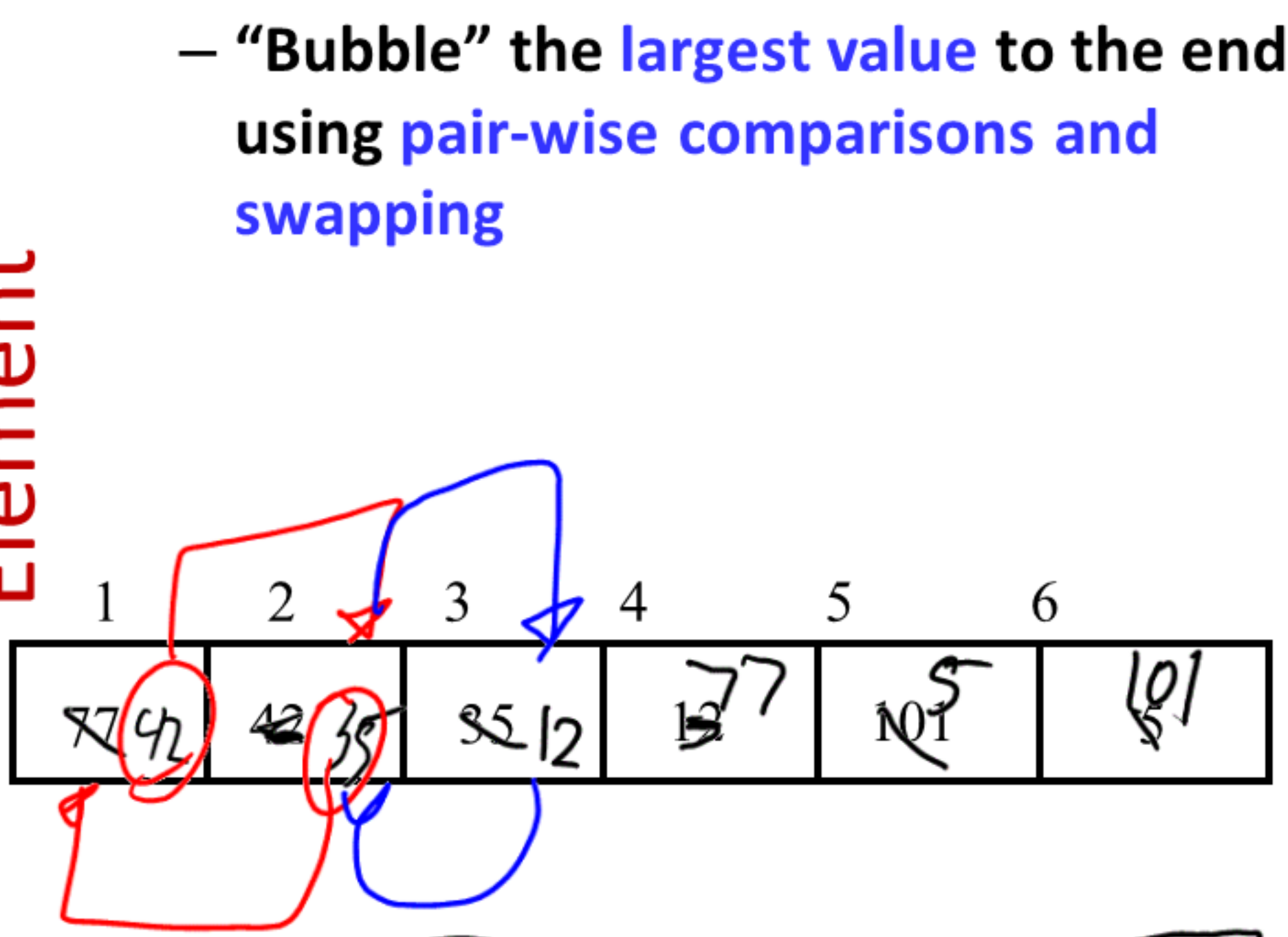
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



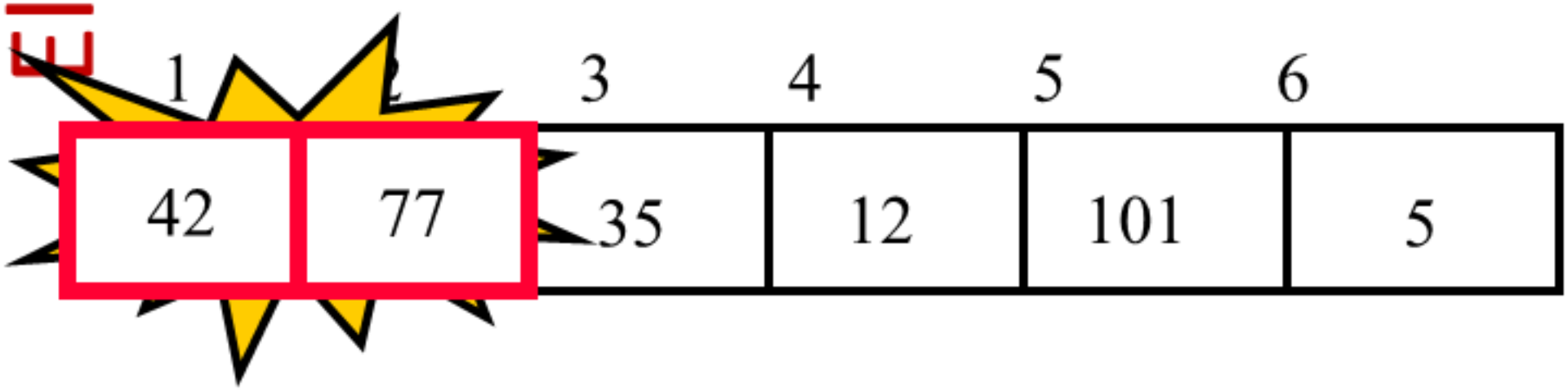
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



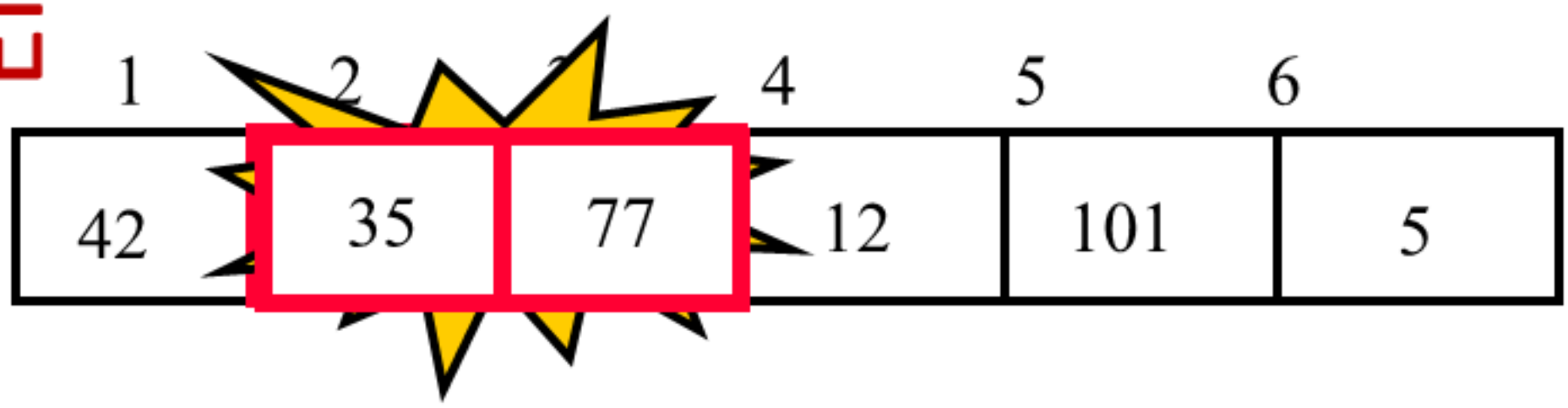
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



The "Bubble Up" Algorithm

```
index <- 1
last_compare_at <- n - 1

loop
  exitif(index >
  last_compare_at)
   if(A[index] > A[index + 1])
   then
      Swap(A[index], A[index + 1])
   endif
   index <- index + 1
endloop
```

Lets write some assembly…