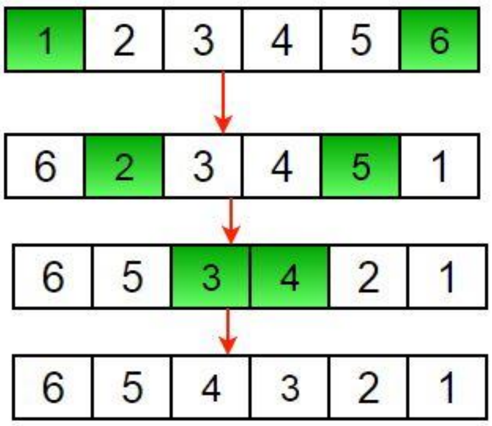
# C programs on Array

1. Write a C program to reverse an array using
    - Recursive function
    - Non-recursive function

    **Algorithm:**

    - Initialize start and end indexes as $start = 0, end = n - 1$
    - In a loop, swap $arr[start]$ with $arr[end]$ and change start and end as follows:

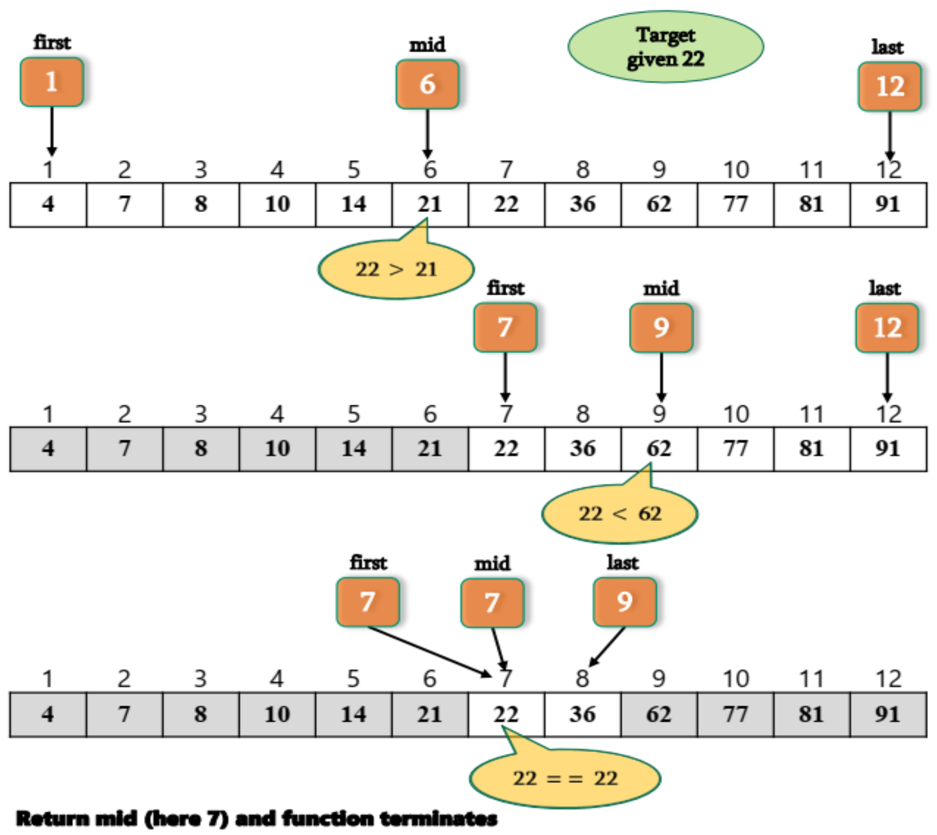
    $$start = start + 1, end = end - 1$$



2. Write a C program to find minimum (or maximum) element in an array.

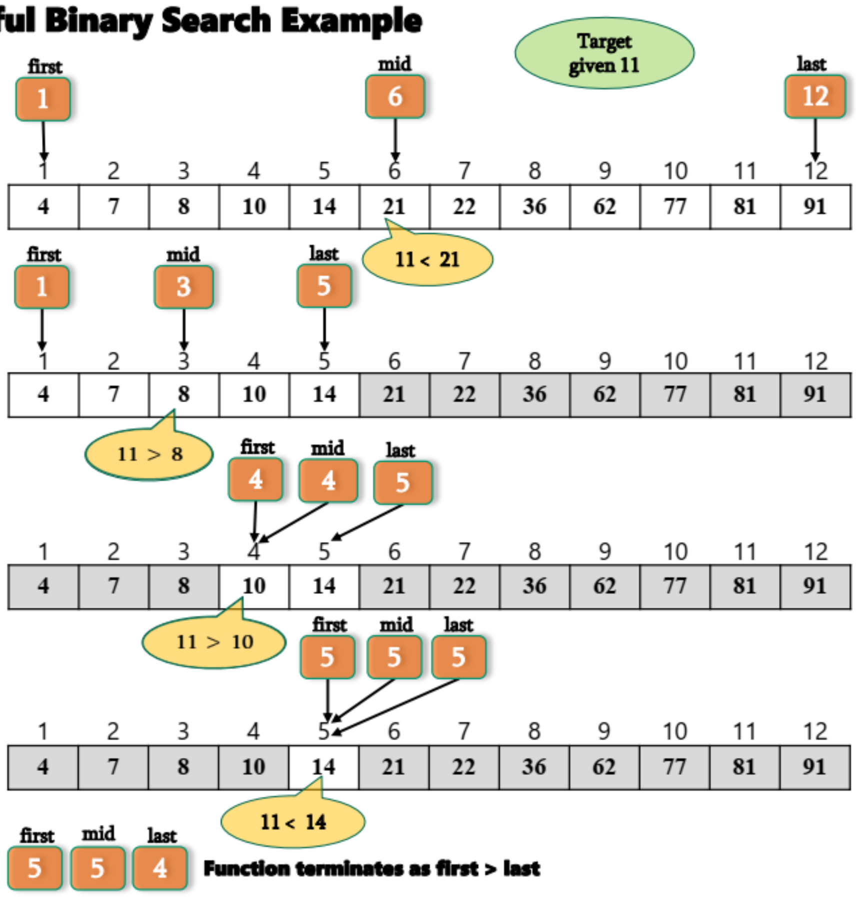
3. **Binary Search** is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to $O(log\ N)$ where $N$ is the array size. Write C program to implement binary search on the following scenario:
    Consider an array $\boldsymbol{arr}[] = \{\mathbf{2, 5, 8, 12, 16, 23, 38, 56, 72, 91}\}$**, and the target = 23**

## **BINARY SEARCH** Example



**Return mid (here 7) and function terminates**

## **Unsuccessful Binary Search Example**



1. Write a C program to implement **Bubble Sort algorithm** on a set of 15 numbers.
   **Algorithm:**
   - traverse the array from left and compare adjacent elements and the higher one is placed at right side.
   - In this way, the largest element is moved to the rightmost end at the first pass.
   - This process is then continued to find the second largest and place it at the last but one position and so on until the whole array is sorted.
   - This algorithm requires (n-1) passes to sort an n-element array.

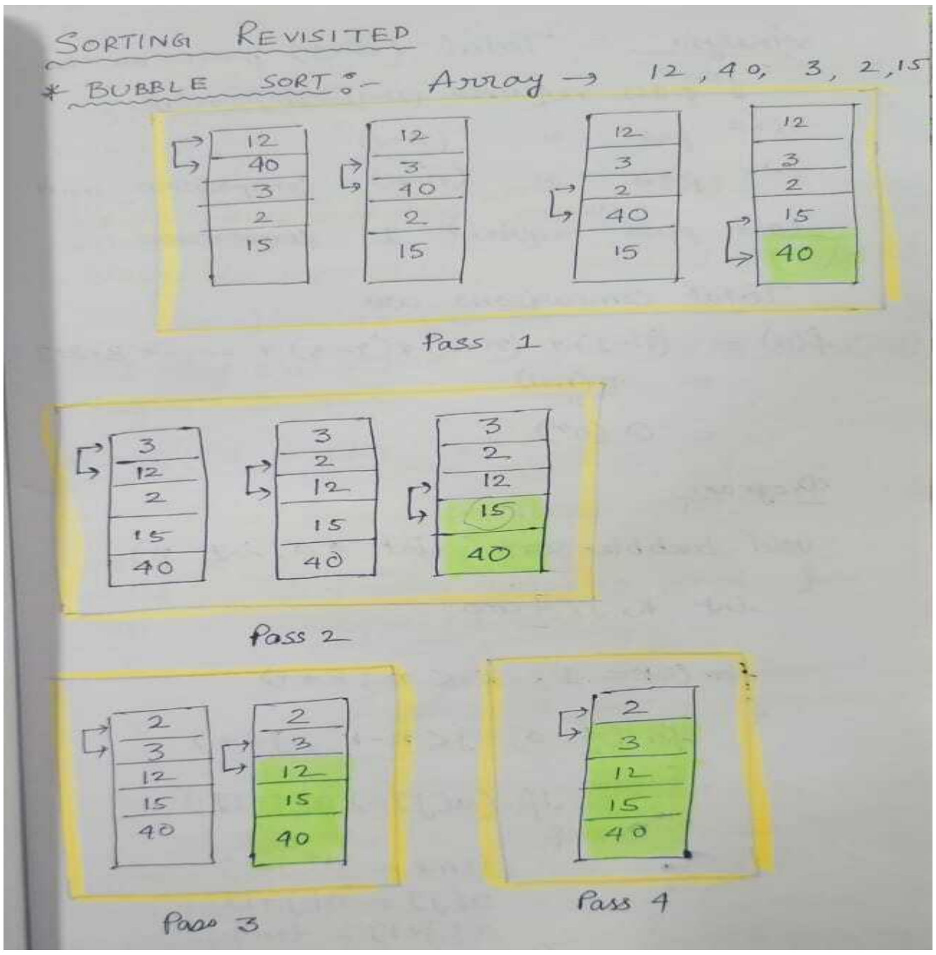
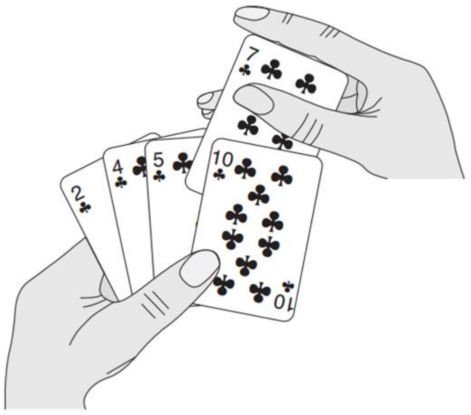
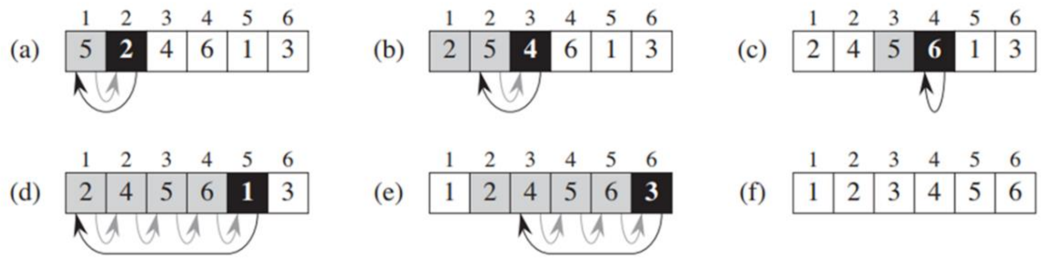


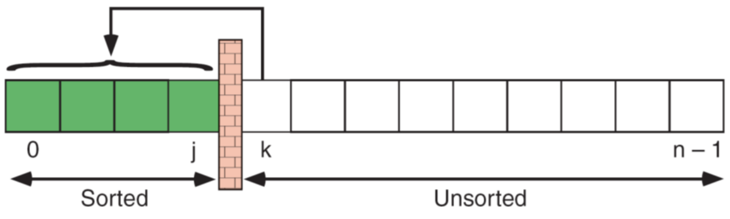
**Fig:** Pictorial representation of Bubble sort algorithm on an array of 5 numbers.

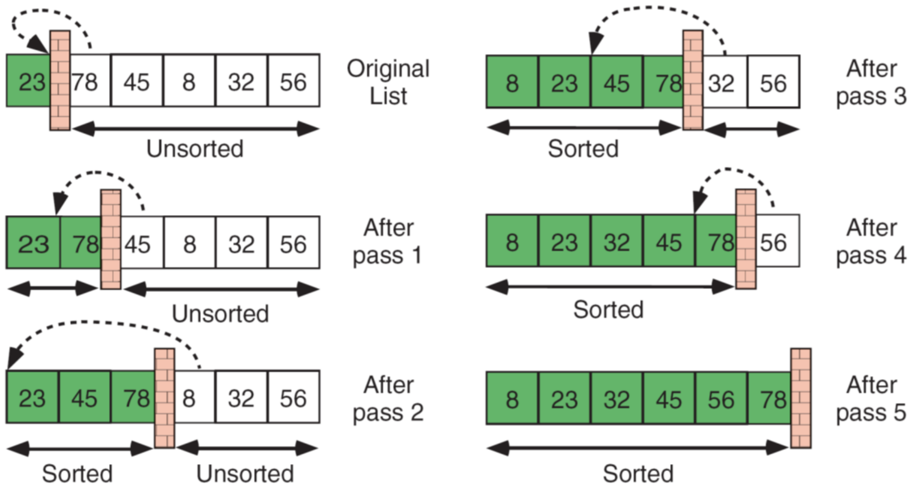
2. Write a C program to implement **Modified Bubble Sort algorithm** on a set of 15 numbers.
   **Algorithm:**
   Bubble sort algorithm possesses a very important property -----------------
   **"Once there is no swapping of elements in a particular pass, there will be no further swapping of elements in the subsequent passes."**
   This property can be exploited to reduce the unnecessary passes. For this purpose we can use a flag variable to determine if any swapping has taken place, if yes only then proceed with the next pass; otherwise stop.

3. Write a C program to implement **Insertion Sort algorithm** on a set of 15 numbers.
   **Algorithm:**
   - Insertion sort works the way many people sort a hand of playing cards.
   - We start with an empty left hand and the cards face down on the table. We then remove one card at a time from the table and insert it into the correct position in the left hand.
   - To find the correct position for a card, we compare it with each of the cards already in the hand, from right to left.









4. Implement **Selection Sort algorithm** on a set of 15 numbers.

**Algorithm:** This algorithm requires (n-1) passes to sort an n-element array.

- **Pass I:**
  Find the location (loc) of the smallest element in the entire array; i.e.,
  $$a[0], a[1], a[2], \dots, a[n-1]$$
  Exchange $a[0]$ with $a[loc]$. Hence, element $a[0]$ is sorted.
- **Pass II:**
  Find the location (loc) of the smallest element in the subarray
  $$a[1], a[2], a[3], \dots, a[n-1]$$
  Exchange $a[1]$ with $a[loc]$. Hence, elements $a[0]$ and $a[1]$ are sorted.

  ………… …………. ………….

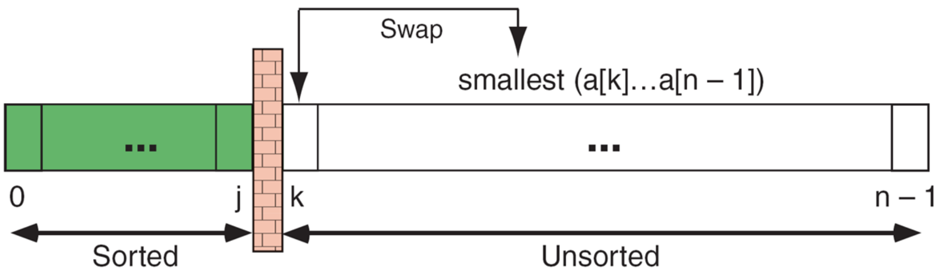
  ………… …………. ………….

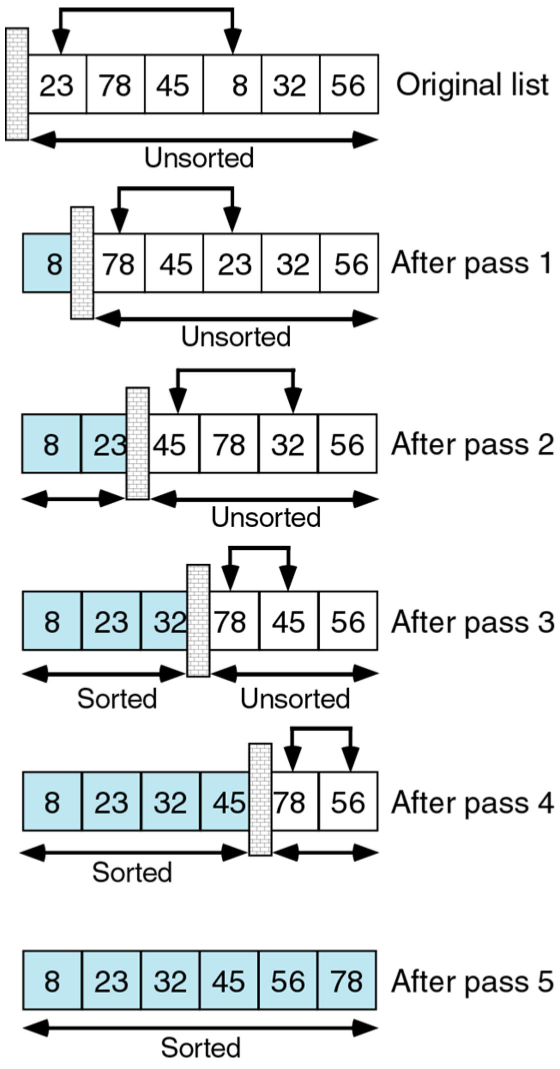
- **Pass k:**
  Find the location (loc) of the smallest element in the subarray
  $$a[k-1], a[k], a[k+1] \ \dots, a[n-1]$$
  Exchange $a[k-1]$ with $a[loc]$. Hence, elements $a[0], a[1], a[2], \dots, a[k-1]$ are sorted.

  ………… …………. ………….

  ………… …………. ………….

- **Pass (n-1):**
  Find the location (loc) of the smallest element in the subarray
  $$a[n-2], a[n-1]$$
  Exchange $a[n-2]$ with $a[loc]$. Hence, elements $a[0], a[1], a[2], \dots, a[n-1]$ are sorted. The whole array is sorted.




23 78 45 8 32 56
Original list
Unsorted
8 78 45 23 32 56
After pass 1
Unsorted
8 23 45 78 32 56
After pass 2
Unsorted
8 23 32 78 45 56
After pass 3
Sorted
Unsorted
8 23 32 45 78 56
After pass 4
Sorted
8 23 32 45 56 78
After pass 5
Sorted