1. Find the maximum element in array

**Pre-condition**: The input array is not null it is included atlas one element.

**Post-condition:** The returned value max is the maximum value in the input array.

**Loop variant**: The loop variable ( i )is decreased by 1 in each iteration of the loop.

**Loop invariant**: The variable max contains the maximum value seen so for the input array ‘arr[0,1,..,i-1]’. The invariant holds true beginning of the loop and is maintained throughout the loop until the end the maximum value in the array.

public static int findMax(int[] arr) {

int max = arr[0];

for (int i = 1; i < arr.length; i++) {

if (arr[i] > max) {

max = arr[i];

}

}

1. Convert the roman numbers to integers

|  |
| --- |
| Symbol I V X L C D M |
|  |
| Value 1 5 10 50 100 500 1000 |

**Pre-condition**: The input should be a valid Roman numeral string consisting of only characters 'I', 'V', 'X', 'L', 'C', 'D', and 'M'.

The input string should not be empty or null.

**Post-condition**: The output should be an integer value representing the equivalent decimal value of the input Roman numeral string.

**Loop variant**: The loop variant is the index variable i, which iterates over the input string from left to right.

**Loop invariant**: The loop invariant is the accumulated sum of the decimal values of the Roman numerals encountered so far during the iteration.

Here's an example Java code for converting a Roman numeral string to an integer using a loop.

public int romanToInt(String s) {

int sum = 0;

int prev = 0;

for (int i = 0; i < s.length(); i++) {

int curr = getValue(s.charAt(i));

if (curr > prev) {

sum -= 2 \* prev;

}

sum += curr;

prev = curr;

}

return sum;

} return result;

}

**3.Move zeros to the end of Array.**

**Pre-condition: The input array numbers is not null.**

**Post-condition: All non-zero elements are at the beginning of the array and all zeros are at the end.**

**Loop invariant: All elements before j are non-zero.**

**Loop variant: The number of zeros encountered so far, which is equal to the current value of j minus the initial value of i.**

public void moveZerosToEnd(int[] nums) {

int i = 0;

int j = 0;

while (j < nums.length) {

if (nums[j] != 0) {

nums[i] = nums[j];

i++;

}

j++;

}

while (i < nums.length) {

nums[i] = 0;

i++;

}

}