1) Return the number of even ints in the given array. Note: the % "mod" operator computes the remainder, e.g. 5 % 2 is 1.

countEvens([2, 1, 2, 3, 4]) → 3  
countEvens([2, 2, 0]) → 3  
countEvens([1, 3, 5]) → 0

LOGIC:-

public int countEvens(int[] nums) {

int count=0;

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

{

if(nums[i]%2==0)

{

count++;

}

}

return count;

}

2) Return the "centered" average of an array of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use int division to produce the final average. You may assume that the array is length 3 or more.

centeredAverage([1, 2, 3, 4, 100]) → 3  
centeredAverage([1, 1, 5, 5, 10, 8, 7]) → 5  
centeredAverage([-10, -4, -2, -4, -2, 0]) → -3

LOGIC:-

public int centeredAverage(int[] nums)

{

int max = nums[0];

int min = nums[0];

int sum = nums[0];

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

{

sum += nums[i];

if(nums[i] > max)

max = nums[i];

else if(nums[i] < min)

min = nums[i];

}

return (sum-max-min) / (nums.length - 2);

}

3) Return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and numbers that come immediately after a 13 also do not count.

sum13([1, 2, 2, 1]) → 6  
sum13([1, 1]) → 2  
sum13([1, 2, 2, 1, 13]) → 6

LOGIC:-

public int sum13(int[] nums) {

int sum=nums[0];

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

if(nums[i]!=30){

sum=sum+nums[i];

}

return (nums[i]+nums[i+1]+nums[i+2]);

}

return 0;

}

4) Given an array of ints, return true if the number of 1's is greater than the number of 4's

more14([1, 4, 1]) → true  
more14([1, 4, 1, 4]) → false  
more14([1, 1]) → true

LOGIC:-

public boolean more14(int[] nums) {

int c1=0;

int c2=0;

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

{

if(nums[i]==1)

{

c1++;

}

else if(nums[i]==4)

{

c2++;

}

}

return c1>c2;

}

5) For this problem, we'll round an int value up to the next multiple of 10 if its rightmost digit is 5 or more, so 15 rounds up to 20. Alternately, round down to the previous multiple of 10 if its rightmost digit is less than 5, so 12 rounds down to 10. Given 3 ints, a b c, return the sum of their rounded values. To avoid code repetition, write a separate helper "public int round10(int num) {" and call it 3 times. Write the helper entirely below and at the same indent level as roundSum().

roundSum(16, 17, 18) → 60  
roundSum(12, 13, 14) → 30  
roundSum(6, 4, 4) → 10

LOGIC:-

public int round10(int num)

{

int digit = num % 10;

if(digit >= 5)

return num + (10 - digit);

return num - digit;

}

public int roundSum(int a, int b, int c)

{ return (round10(a) + round10(b) + round10(c)); }