1. Define the n-based integer rounding of an integer k to be the nearest multiple of n to k. If two multiples of n are equidistant use the greater one. For example the 4-based rounding of 5 is 4 because 5 is closer to 4 than it is to 8, the 5-based rounding of 5 is 5 because 5 is closer to 5 that it is to 10, the 4-based rounding of 6 is 8 because 6 is equidistant from 4 and 8, so the greater one is used, the 13-based rounding of 9 is 13, because 9 is closer to 13 than it is to 0,

   Write a function named **doIntegerBasedRounding** that takes an integer array and rounds all its positive elements using n-based integer rounding. A negative element of the array is not modified and if n <=0, no elements of the array are modified. Finally you may assume that the array has at least two elements. Hint: In integer arithmetic, (6/4) * 4 = 4

   The function signature is void **doIntegerBasedRounding(int[ ] a, int n)** where n is used to do the rounding

| **if a is** | **and n is** | **then a becomes** | **reason** |
|---|---|---|---|
| {1, 2, 3, 4, 5} | 2 | {2, 2, 4, 4, 6} | because the 2-based rounding of 1 is 2, the 2-based rounding of 2 is 2, the 2-based rounding of 3 is 4, the2-based rounding of 4 is 4, and the 2-based rounding of 5 is 6. |
| {1, 2, 3, 4, 5} | 3 | {0, 3, 3, 3, 6} | because the 3-based rounding of 1 is 0, the 3-based roundings of 2, 3, 4 are all 3, and the 3-based rounding of 5 is 6. |
| {1, 2, 3, 4, 5} | -3 | {1, 2, 3, 4, 5} | because the array is not changed if n <= 0. |
| {-1, -2, -3, -4, -5} | 3 | {-1, -2, -3, -4, -5} | because negative numbers are not rounded |

2. A number n>0 is called cube-powerful if it is equal to the sum of the cubes of its digits. Write a function named isCubePowerful that returns 1 if its argument is cube-powerful; otherwise it returns 0. The function prototype is int isCubePowerful(int n);
Hint: use modulo 10 arithmetic to get the digits of the number.

Examples:

| if n is | return | because |
|---|---|---|
| 153 | 1 | because $153 = 1^3 + 5^3 + 3^3$ |
| 370 | 1 | because $370 = 3^3 + 7^3 + 0^3$ |
| 371 | 1 | because $371 = 3^3 + 7^3 + 1^3$ |
| 407 | 1 | because $407 = 4^3 + 0^3 + 7^3$ |
| 87 | 0 | because $87 \mathrel{!}= 8^3 + 7^3$ |
| 0 | 0 | because n must be greater than 0. |
| -81 | 0 | because n must be greater than 0. |

3. An array is zero-plentiful if it contains at least one 0 and every sequence of 0s is of length at least 4. Write a method named **isZeroPlentiful** which returns the number of zero sequences if its array argument is zero-plentiful, otherwise it returns 0. If you are programming in Java or C#, the function signature **is int isZeroPlentiful(int[ ] a)** .The function signature is int isZeroPlentiful(int a[ ], int len) where len is the number of elements in the array a.
Example

| a is | then function returns | reason |
|---|---|---|
| {0, 0, 0, 0, 0}1 | 1 | because there is one sequence of 0s and its length >= 4. |
| {1, 2, 0, 0, 0, 0, 2, -18, 0, 0, 0, 0, 0, 12}1 | 2 | because there are two sequences of 0s and both have lengths >= 4. |
| {0, 0, 0, 0, 1, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0}1 | 3 | because three are three sequences of zeros and all have length >=4 |
| {1, 2, 3, 4}1 | 0 | because there must be at least one 0. |
| {1, 0, 0, 0, 2, 0, 0, 0, 0} | 0 | because there is a sequence of zeros whose length is less < 4. |
| {0} | 0 | because there is a sequence of zeroes whose length is < 4. |

4. An array is called zero-balanced if its elements sum to 0 and for each positive element n, there exists another element that is the negative of n. Write a function named **isZeroBalanced** that returns 1 if its argument is a zero-balanced array. The function signature is **int isZeroBalanced(int[ ] a)**

Examples:

| **if a is** | **return** |
|---|---|
| {1, 2, -2, -1} | 1 because elements sum to 0 and each positive element has a corresponding negative element. |
| {-3, 1, 2, -2, -1, 3} | 1 because elements sum to 0 and each positive element has a corresponding negative element. |
| {3, 4, -2, -3, -2} | 0 because even though this sums to 0, there is no element whose value is -4 |
| {0, 0, 0, 0, 0, 0} | 1 this is true vacuously; 0 is not a positive number |
| {3, -3, -3} | 0 because it doesn't sum to 0. (Be sure your function handles this array correctly) |
| {3} | 0 because this doesn't sum to 0 |
| {} | 0 because it doesn't sum to 0 |