Write an Efficient Method to Check if a Number is Multiple of 3

The very first solution that comes to our mind is the one that we learned in school. If sum of digits in a number is multiple of 3 then number is multiple of 3 e.g., for 612 sum of digits is 9 so it's a multiple of 3. But this solution is not efficient. You have to get all decimal digits one by one, add them and then check if sum is multiple of 3.

There is a pattern in binary representation of the number that can be used to find if number is a multiple of 3. If difference between count of odd set bits (Bits set at odd positions) and even set bits is multiple of 3 then is the number.

Example: 23 (00..10111)
1) Get count of all set bits at odd positions (For 23 it's 3).
2) Get count of all set bits at even positions (For 23 it's 1).
3) If difference of above two counts is a multiple of 3 then number is also a multiple of 3.

(For 23 it's 2 so 23 is not a multiple of 3)

Take some more examples like 21, 15, etc…

```
Algorithm: isMutlipleOf3(n)
1) Make n positive if n is negative.
2) If number is 0 then return 1
3) If number is 1 then return 0
4) Initialize: odd_count = 0, even_count = 0
5) Loop while n != 0
    a) If rightmost bit is set then increment odd count.
    b) Right-shift n by 1 bit
    c) If rightmost bit is set then increment even count.
    d) Right-shift n by 1 bit
6) return isMutlipleOf3(odd_count - even_count)
```

Proof:

Above can be proved by taking the example of 11 in decimal numbers. (In this context 11 in decimal numbers is same as 3 in binary numbers)
If difference between sum of odd digits and even digits is multiple of 11 then decimal number is multiple of 11. Let's see how.

Let's take the example of 2 digit numbers in decimal
AB = 11A -A + B = 11A + (B – A)
So if (B – A) is a multiple of 11 then is AB.

Let us take 3 digit numbers.

ABC = 99A + A + 11B – B + C = (99A + 11B) + (A + C – B)
So if (A + C – B) is a multiple of 11 then is (A+C-B)

Let us take 4 digit numbers now.
ABCD = 1001A + D + 11C – C + 999B + B – A
= (1001A – 999B + 11C) + (D + B – A -C)
So, if (B + D – A – C) is a multiple of 11 then is ABCD.

This can be continued for all decimal numbers.
Above concept can be proved for 3 in binary numbers in the same way.

Time Complexity: O(logn)

Program:

```
#include<stdio.h>

/* Fnction to check if n is a multiple of 3*/
int isMultipleOf3(int n)
{
    int odd_count = 0;
    int even_count = 0;

    /* Make no positive if +n is multiple of 3
       then is -n. We are doing this to avoid
       stack overflow in recursion*/
    if(n < 0)   n = -n;
    if(n == 0) return 1;
    if(n == 1) return 0;
```

```
    while(n)
    {
        /* If odd bit is set then
           increment odd counter */
        if(n & 1)
           odd_count++;
        n = n>>1;

        /* If even bit is set then
           increment even counter */
        if(n & 1)
            even_count++;
        n = n>>1;
    }

     return isMultipleOf3(abs(odd_count - even_count));
}

/* Program to test function isMultipleOf3 */
int main()
{
    int num = 23;
    if (isMultipleOf3(num))
        printf("num is multiple of 3");
    else
        printf("num is not a multiple of 3");
    getchar();
    return 0;
}
```