

## 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.

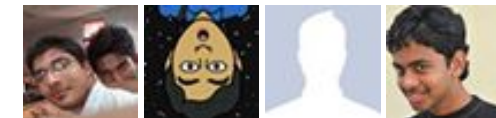
Google™ Custom Search



GeeksforGeeks



53,526 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

```

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 isMultipleOf3(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:**  $O(\log n)$

**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;

```

# HP Chromebook 11

 [google.com/chromebook](https://google.com/chromebook)

Everything you need in one laptop.  
Made with Google. Learn more.



## Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

```

/* 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;
}

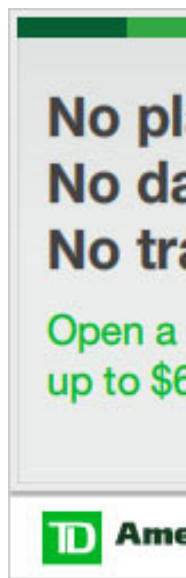
```

# Deploy Early. Deploy Often.

DevOps from  
Rackspace:

## Automation

**FIND OUT HOW ►**



705



Subscribe

## Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 20 minutes ago

**Aman** Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 59 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

**Sanjay Agarwal** bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

**GOPI GOPINATH** @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

**newCoder3006** If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

## Related Topics:

- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)
- Find position of the only set bit
- Swap all odd and even bits
- Add two bit strings
- Write your own strcmp that ignores cases
- Binary representation of a given number



3

Tweet

0

4

Writing code in comment? Please use [ideone.com](http://ideone.com) and share the link here.

46 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion



with the algorithm...



**iamcoded** • 2 months ago

can any one explain how is complexity  $O(\log n)$ ?

^ | v .



**Rohan** • 9 months ago

```
#include <stdio.h>
#include <math.h>
int main()
{
    int i=0,n, p = 0, d, n_original;
    printf("Enter number : ");
    scanf("%d",&n)
    printf("Enter Diviser : ");
    scanf("%d",&d)

    n_original = n;

    while(n>1) {
        i++;
        n>>=1;
    }
    n=pow(2,i);

    while(n)
```

see more

1 ^ | v .



**Pankaj Goyal** • 10 months ago

I don't understand why it is more efficient....for a thousand digit no...by fir and checking if it is divisible by 3, in that if I would take 1000 iterations to solve

AdChoices

[▶ Numbers Number](#)

[▶ Number System](#)

[▶ Code Number](#)

AdChoices

[▶ Numbers Number](#)

[▶ Code Number](#)

[▶ Math Number](#)

AdChoices

[▶ Even Number](#)

[▶ Number Sense](#)

[▶ Number of First](#)

log(9999.....1000 times) iterations to solve it...  
so how is it more efficient?

1 ^ | v .



**jugal** • 10 months ago

complexity of method 4 is not  $\log(n)$ . because we are performing constant operations. please check.

^ | v .



**jugal** → jugal • 10 months ago

sorry misplaced previous comment as it is for next-power-of-2 post.

^ | v .



**abhishek08aug** • a year ago

Why can't we just take modulo of number with 3 and if it comes out to be a zero multiple of 3?

Why so much of unnecessary extra mind work?

^ | v .



**abhishek08aug** → abhishek08aug • a year ago

Probably because taking modulo is  $O(n/3)$ ?

^ | v .



**lakshay** → abhishek08aug • 11 months ago

Modulo involves division which takes  $O(m)$  (atleast theoretically).  
When a no is a power of 2, the modulo internally does it by bit-wise operations.  
efficient!  
correct me if i am wrong.

^ | v .



**Kushagra Sinha** • a year ago



@admin Correct me if I am wrong but the complexity of the posted solution is

In the first step of recursion, we need  $\log(n)$  steps (since there are  $\log_2(n)$  num  
Then further down the recursion chain, the complexity in \*worst\* case will be  $\log_2(\log_2(n))$  digits) and then  $\log(\log(\log n))$  and so on. Therefore the complex

Let us be sloppy for a second and assume the complexity is  $O(\log n)$  which is

Now, let us consider the "school" algorithm. It sums the digits of  $n$  in its decimal  
above, its complexity will be  $\log_{10}(n)$ .

Since,  $\log_{10}(n) < \log_2(n)$  as  $n$  tends to infinity, the "school" algorithm is more e

Common sense explanation: Since we are looking at each digit anyhow, it makes no sense to convert to base 2 so that the number of digits considered is less. Therefore a solution in base

Please clarify.

1 ^ | v .



**Akhil** → Kushagra Sinha · 11 months ago

base 2 solution involves checking bits of the number only.

If we find the digits and sum them, it will take much more time.

```
/* Paste your code here (You may delete these lines if not write your code here)
^ | v .
```



**HLS.nirma** · 2 years ago

The code can be optimized here:

```
if(n == 0) return 1;
if(n == 1) return 0;
```

```
/*Improvement */
```

```
if(n == 0) return 1;
if(n <= 2) return 0;
/* One function call will be less. */
```

1 ^ | v .



**Amit** → HLS.nirma · 8 days ago

then it ll return 0 if n=0 .

^ | v .



**Arpitha** · 2 years ago

The simple way of doing that will be , convert the given number into string using given number, check whether sum is 3 6 or 9 , if yes then its divisible :)

^ | v .



**AG** → Arpitha · a year ago

Brilliant !

^ | v .



**pritybhudolia** → AG · a year ago

@Arpitha

I am not sure whether i got u or not. But if u mean

42= 4+2=6 hence divisible by 3

12=1+2=3 hence divisible by 3

then wat about 39 as 3+9=12

if u ask again to add until it becomes single digit, then i think sin

^ | v .



**Prateek** · 2 years ago

There is a small error in explanation:

Instead of:



instead of.

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.

it should be:

$$ABCD = 1001A + 99B + 11C - A + B - C + D$$

$$ABCD = (1001A + 99B + 11C) + (D + B - A - C)$$

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | v .



**Dhaval Patel** · 2 years ago

[sourcecode language="java"]

```
public static void main(String[] args) {
```

```
int number = 23;
```

```
int andOperation = 0;
```

```
int oddCount = 0;
```

```
int evenCount = 0;
```

```
while (number != 0) {
```

```
andOperation = number & 1;
```

```
if (andOperation == 1) {
```

```
oddCount++;
```

```
}
```

```
number = number >>> 1;
```

```
andOperation = number & 1;
```

```
if (andOperation == 1) {
```

```
evenCount++;
```

see more

^ | v .



Red Lv · 2 years ago

I think the idea DFA would work perfectly here with the time complexity  $O(\lg n)$ ,

```
int rem=0;
```

```
while(n)
```

```
{
```

```
int bit=n&1;
```

```
rem=(rem*2+bit)%3;
```

```
n=n>>1;
```

```
}
```

```
return (rem==0);
```

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | v .



shal → Red Lv · 2 years ago

like it

```
/* Paste your code here (You may delete these lines if not wr
```

^ | v .



praveen Raj · 3 years ago

```
int checkdiv(int num)
```

```
{
```

```

int n;
aim:
while(num!=0)
{
    n+=(num&0x0f);
    num>>=4;
}
if((n>>4)!=0)
{
    num=n;
    goto aim;
}
else
{
    if((n==0)|| (n==3)|| (n==6)|| (n==9))
        return 1;
    else
        return 0;
}
}

```

^ | v .



**KK123** · 3 years ago

Here's a clean and working code:

```

#include<iostream>
using namespace std;

int bitCount(int n)
{
    int count = 0;
    while(n)
    {

```

```

        count++;
        n = n & (n - 1);
    }
    return count;
}

int main()
{

```

see more



**MD03** → KK123 · 11 months ago

liked it!!



**john** · 3 years ago

```

boolean isMultipleOfThree(int n) {

    boolean oddBit = true;
    int diff = 0;
    for (; n > 0; n >>= 1, oddBit = !oddBit) {
        diff += oddBit ? n & 1 : -(n & 1);
    }
    return diff % 3 == 0;
}

```



**john** → john · 3 years ago

```

boolean isMultipleOfThree(int n) {

```

```

    n = Math.abs(n);

```

```

    n = Math.abs(n),
    boolean oddBit = true;
    int diff = 0;
    for (; n > 0; n >>= 1, oddBit = !oddBit) {
        diff += oddBit ? n & 1 : -(n & 1);
    }
    return diff % 3 == 0;
}

```

^ | v .



**varnika** · 3 years ago

can you generalize divisibility by any no.?

^ | v .



**atiq** → varnika · a year ago

Not possible...

```

/* Paste your code here (You may delete these lines if not wr

```

^ | v .



**atiq** → atiq · a year ago

I mean no better method.....

^ | v .



**Frank** · 3 years ago

can someone explain (In this context 11 in decimal numbers is same as 3 in b

^ | v .



**arnav.agarwal** · 4 years ago

Also:

```

#include<stdio.h>

int remainderby3(unsigned int i) { /* good for 32-bit unsigned int */
    i = (i & 65535) + (i >> 16);
    i = (i & 65535) + (i >> 16);
    i = (i & 255) + (i >> 8);
    i = (i & 255) + (i >> 8);
    i = (i & 15) + (i >> 4);
    i = (i & 15) + (i >> 4);
    i = (i & 3) + (i >> 2);
    i = (i & 3) + (i >> 2);
    return (i != 3)? i : 0;
}

void main()
{
    int n;
    scanf("%d", &n);
    int a= remainderby3(n);
    if(a==0)
        printf("Divisible by 3");
    else
        printf("not divisible by 3");
}

```

^ | v .



tech.login.id2 · 4 years ago

@Geeks4Geeks:

Please provide a proof of this theory too.

A derivation showing why a number would be divisible by 3 if diff in its odd and there before we accept its correctness.

@game:

Please elaborate a little what you mean. I could not understand anything.

^ | v ·



**pennypecker** · 4 years ago

The quotient for 'B' should be 99 in:

$$\begin{aligned} \text{"ABCD"} &= 1001A + D + 11C - C + 999B + B - A \\ &= (1001A - 999B + 11C) + (D + B - A - C) \end{aligned}$$

^ | v ·



**sunny321** · 4 years ago

please comment on soln of "game"..

^ | v ·



**game** · 4 years ago

Hello, I would like to extend the question to finding the divisibility of any number operator (its VERY costly).

The following code does the same, hope u enjoy the beauty of the algorithm :)

```
#include<stdio.h>
int main()
{
    int n, p = 0, d, t;
    scanf("%d%d",&n,&d); /* n is number and d is divisor */
    t = n;
    n--;
    n |= n>>1;
    n |= n>>2;
    n |= n>>4;
    n |= n>>8;
    n |= n>>16;
```

```
n++;  
if ( (t&(t-1) ) != 0)
```

[see more](#)

^ | v ·



**Rohan** → game · 9 months ago

Perhaps a more straight forward way to point at MSB.

```
#include <stdio.h>  
#include <math.h>  
int main()  
{  
    int i=0,n, p = 0, d, n_original;  
    printf("Enter number : ");  
    scanf("%d",&n);  
    printf("Enter Diviser : ");  
    scanf("%d",&d);  
  
    n_original = n;  
  
    while(n>1) {  
        i++;  
        n>>=1;  
    }  
}
```

[see more](#)

^ | v ·



**kas** → game · 3 years ago

Great work :)

^ | v ·





**Surabhi** → game · 3 years ago

A slight correction to the code.

while(n) is an infinite loop here.

it should be:

```
while ( n )
{
    p = (n&t)?( (p<<1) + 1):(p<=<1);
    p = p>=d?p-d:p;
    n>>=1;
}
```



**mannirulz** → game · 4 years ago

this doesnt seems to be working... can u check with values(27 4)



**geeksforgeeks** · 5 years ago

@Arun: Your solution is correct. While you are using a construct provided by p implemented an algorithm for multiple of 3.

Time complexity of % operator will also be  $O(\log n)$  as it is also dependent on t n.



**tiger** → geeksforgeeks · 2 years ago

Could you please explain me how it is  $O(\log n)$  in the method you have Please...



**candis** → [geeksforgeeks](#) · 3 years ago

i prbably m not getting how is it working for the cases lyk, 1000(8)  
even count=1  
odd count=0

remainder -1 absolute(-1)=1  
but the expected answer was 2....

^ | v ·



**Arun** · 5 years ago

I am not sure i am right

```
if Num%3==0 {print("Num is multiple of 3")}  
else{print("Num is not multiple of 3")}
```

Time complexity : O(1)

^ | v ·



**Devender Rao** → [Arun](#) · 2 months ago

missed the whole point. here we are not taking just about the % operat

^ | v ·



**vishal.mnnit** → [Arun](#) · 4 years ago

u r rt

^ | v ·



**geeksforgeeks** · 5 years ago

@Joe: It works for 6 and 9 also. For 6 (110), difference of set bit count at even of 3). Same is true for 9. You can verify it by replacing 23 with 3 or 6 in the abc

^ | v ·



**kg1020** → [geeksforgeeks](#) · 2 years ago



binary of 21 is 00...010101  
so how it will work for 21 ?  
plse explain...

^ | v .



**joe blow** · 5 years ago

Does that algorithm work? It doesn't work for numbers like 6 or 9

^ | v .



Subscribe



Add Disqus to your site

