

Check for Integer Overflow

Write a “C” function, `int addOvf(int* result, int a, int b)` If there is no overflow, the function places the resultant = sum `a+b` in “result” and returns 0. Otherwise it returns -1. The solution of casting to long and adding to find detecting the overflow is not allowed.

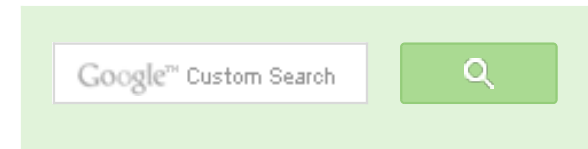
Method 1

There can be overflow only if signs of two numbers are same, and sign of sum is opposite to the signs of numbers.

- 1) Calculate sum
- 2) If both numbers are positive and sum is negative then return -1
Else
If both numbers are negative and sum is positive then return -1
Else return 0

```
#include<stdio.h>
#include<stdlib.h>

/* Takes pointer to result and two numbers as
arguments. If there is no overflow, the function
places the resultant = sum a+b in “result” and
returns 0, otherwise it returns -1 */
int addOvf(int* result, int a, int b)
{
    *result = a + b;
    if(a > 0 && b > 0 && *result < 0)
        return -1;
    if(a < 0 && b < 0 && *result > 0)
        return -1;
    return 0;
}
```



GeeksforGeeks



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



Facebook

Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Geometric Algorithms

```

int main()
{
    int *res = (int *)malloc(sizeof(int));
    int x = 2147483640;
    int y = 10;

    printf("%d", addOvf(res, x, y));

    printf("\n %d", *res);
    getchar();
    return 0;
}

```

Time Complexity : O(1)

Space Complexity: O(1)

Method 2

Thanks to Himanshu Aggarwal for adding this method. This method doesn't modify *result if there is an overflow.

```

#include<stdio.h>
#include<limits.h>
#include<stdlib.h>

int addOvf(int* result, int a, int b)
{
    if( a > INT_MAX - b)
        return -1;
    else
    {
        *result = a + b;
        return 0;
    }
}

int main()
{
    int *res = (int *)malloc(sizeof(int));
    int x = 2147483640;
    int y = 10;

    printf("%d", addOvf(res, x, y));
    printf("\n %d", *res);
}

```

HP Chromebook 11

 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](#)

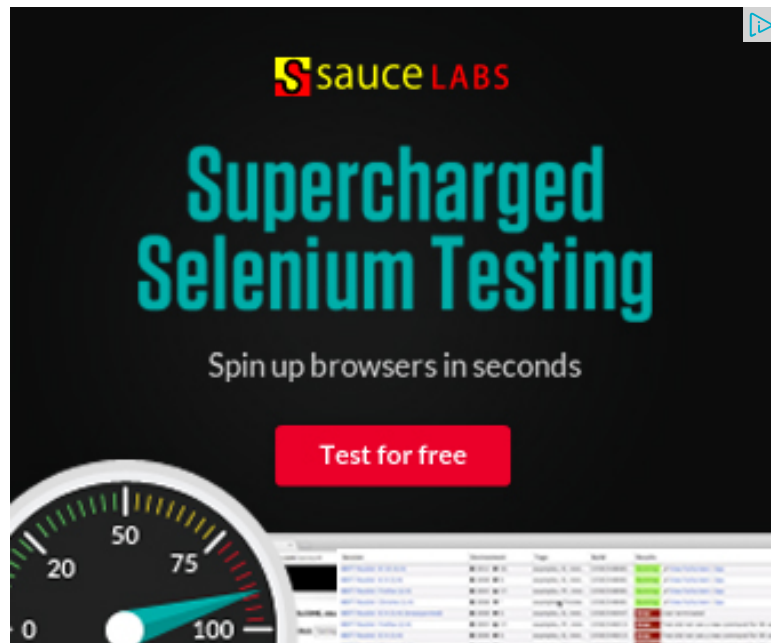
[Sorted Linked List to Balanced BST](#)

```
getchar();  
return 0;  
}
```

Time Complexity : $O(1)$

Space Complexity: $O(1)$

Please write comments if you find any bug in above codes/algorithms, or find other ways to solve the same problem



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



1



Tweet

1



0



705



Subscribe

Writing code in comment? Please use ideone.com and share the link here.

21 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...

**neelabhsingh** · 6 months ago

How to check the overflow without using branch?

^ | v ·

**amit** · 7 months ago

Even 1st method won't work if we take $a > 0$ at boundary level and $b < 0$ in that c
0

^ | v ·

**Anubhav Gupta** · 9 months ago

I think both the methods are partially correct.

If one(or both) of the given numbers are already exceeding the integer ranges,
it.

For eg.

if $a = 32768$, $b = 3$

then result = -32765

Is that wrong?

^ | v ·

**ajith** · 9 months ago

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

AdChoices

[▶ Programming C++](#)[▶ Long Int C++](#)[▶ Convert Int](#)



```
int addOvf(int *result, int a,int b)
{
    int temp;
    *result=a+b;
    temp=(b-a)/2 + a;
    if(*result/2 == temp)
        return 0;

    return -1;
}
```

^ | v .



akshat gupta · 11 months ago

```
msb1=n1&(1<<31)
msb2=n2&(1<<31)
n1=n1-(msb1<<31);
n2=n2-(msb2<<31);

overflow= (n1+n2)&(1<<31)
if((msb1&&msb2)||((msb1||msb2)&&overflow )
return(-1);
return(0);
```

^ | v .



ultimate_coder · a year ago

hey...plzz tell me... is method 1 using roll over concept???

^ | v .



Hanish · a year ago

AdChoices

[▶ Int To](#)

[▶ Overflow](#)

[▶ Positive Integer](#)

AdChoices

[▶ Long Integer](#)

[▶ Int Byte Array](#)

[▶ Int Bits](#)



Method 2 does not work for the case when b is negative.

e.g.

if a= 50 and b= -10

result should be 40 with no overflow.

but method 2 gives the ans as -1

since, $\text{INT_MAX} - b = -2147483639$

thus, $a > \text{INT_MAX} - b$ hence it returns -1 which is incorrect.

Kindly correct this

^ | v .



RDD → Hanish · a year ago

Yes you are correct.

^ | v .



pajju · 3 years ago

```
/* portable for any Compiler*/
int addOvf(int* result, int a, int b)
{
    if(a>0 && b>0)
        if ((a+b)<0)                //a+b<0 at Positive Overflow
            return -1;
    else if(a<0 && b<0)
        if ((a+b)>0)                //a+b>0 at Negative Overflow
            return -1;

    else
    {
        *result = a+b;
        return 0;
    }
}
```

^ | v .



dipendra · 3 years ago

this site describes the overflow detection to the best possible in generalized a

^ | v .



Himanshu Aggarwal · 4 years ago

Hi,

I think that first solution is non-compliant.

K&R, second edition, Appendix A- Section A.7, quotes :

"The handling of overflow, divide check, and other exceptions in expression ev language."

Hence the solution is non-portable as per my understanding.

Regards,
Himanshu Aggarwal

^ | v .



pi → Himanshu Aggarwal · 6 months ago

if((a>INT_MAX-b)||a<int_min-b)) i="" think="" this="" statement="" shou
if(="" a=""> INT_MAX - b) statement of second method

^ | v .



pajju → Himanshu Aggarwal · 3 years ago

Your solution is not portable. You are expecting INT_MAX to be defined

^ | v .



Bandicoot · 4 years ago



I didn't understand the motivation behind method 2. The post says method 2 is *result. In that case why can't we use temp = a + b in method 1 instead of *res. Also, in method 2, is there an assumption that both a > 0 and b > 0 ? If both are another case which checks a < INT_MIN - b ?

^ | v .



Himanshu Aggarwal → Bandicoot · 4 years ago

Hi,

That would be an underflow if a < 0 and b < 0. The aforementioned problem detection.

Thanks,
Himanshu

^ | v .



GeeksforGeeks · 4 years ago

@Himanshu Aggarwal: Thanks for suggesting a new method. We have included it.

@Hary: If we do not want to modify *result in case of overflow then we can use the method suggested by Himanshu Aggarwal.

^ | v .



Himanshu Aggarwal · 4 years ago

Hi,

Let the two numbers be 'a' and 'b'

we can also write the overflow condition as :

```
int addOvf(int* result, int a, int b)
{
    if( a > INT_MAX - b)
```



```
    else {  
        *result = a + b;  
        return 0;  
    }  
}
```

^ | v .



Hary · 4 years ago

Not very sure but to me the main solution also seems to be non-compliant. Re itself says store the sum in "result" only if an overflow has not occurred.

^ | v .



geeksforgeeks · 5 years ago

@Raj: Thanks, your solution looks concise, but when looked into this carefully It always returns 0.

^ | v .



Gautam · 5 years ago

Raj's solution doesn't work for boundary conditions.

^ | v .



raj · 5 years ago

```
int addOvf(int* result, int a, int b)  
{  
    *result = a + b;  
    if((*result - a) == b)  
        return 0;  
  
    return -1;  
}
```

 [Subscribe](#)

 [Add Disqus to your site](#)

@geeksforgeeks, **Some rights reserved**

[Contact Us!](#)

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team