Error in Competitive programming Online Code

Runtime error:

Segmentation fault:
Override memory and running to fill more memory.

output limit exceeded:
Your program has printed too much data to output.

floating point error:
This usually occurs when you're trying to divide a number by 0, or trying to take the square root of a negative number.

NZEC
(non-zero exit code)

1. Trying to allocate too much memory during code execution may also be one of the reasons.
2. It could happen if your program threw an exception which was not caught.
3. this message means that the program exited returning a value different from 0 to the shell.

##

Geeks MMI (Multiplicative inverse): do by making table finding q and r and the t=t1-qt2

z = (x/y) % M;
instead we should perform
y_inv = findMMI(y, M);
z = (x * y_inv) % M;

##

**BEST LINK FOR ALL BITWISE OPERATION (LEETCODE)

https://leetcode.com/problems/sum-of-two-integers/discuss/84278/A-summary:-how-to-use-bit-manipulation-to-solve-problems-easily-and-efficiently

@@BIt Maniputation@@https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/tutorial/

SUM WITHOUT + operator

```
#include<stdio.h>

int Add(int x, int y)
{
    // Iterate till there is no carry
    while (y != 0)
    {
        // carry now contains common set bits of x and y
        int carry = x & y;

        // Sum of bits of x and y where at least one of the bits is not set
        x = x ^ y;

        // Carry is shifted by one so that adding it to x gives the required sum
        y = carry << 1;
    }
    return x;
}

int main()
{
    printf("%d", Add(15, 32));
    return 0;
}
```

x^y -> it gives common digits.

x&y -> carry is the common set of bits and it is suited to left by 1 so that when we take xor it will give exact sum.

Reverse actual bits of the given number

Input : 11
Output : 13
(11)10 = (1011)2.
After reversing the bits we get:
(1101)2 = (13)10.

```
// C++ implementation to reverse bits of a number
#include <bits/stdc++.h>

using namespace std;

// function to reverse bits of a number
unsigned int reverseBits(unsigned int n)
{
    unsigned int rev = 0;

    // traversing bits of 'n' from the right
    while (n > 0)
    {
        // bitwise left shift
        // 'rev' by 1
        rev <<= 1;

        // if current bit is '1'
        if (n & 1 == 1)
            rev ^= 1;

        // bitwise right shift
        // 'n' by 1
        n >>= 1;

    }

    // required number
    return rev;
}

// Driver program to test above
int main()
{
    unsigned int n = 11;
    cout << reverseBits(n);
    return 0;
}
```

1) How to check if a given number is a power of 2 ?

The same problem can be solved using bit manipulation. Consider a number x that we need to check for being a power for 2. Now think about the binary representation of (x-1). (x-1) will have all the bits same as x, except for the rightmost 1 in x and all the bits to the right of the rightmost 1.

Let, $x = 4 = (100)_2$

$x - 1 = 3 = (011)_2$

Let, x = 6 = $(110)_2$

x - 1 = 5 = $(101)_2$

It might not seem obvious with these examples, but binary representation of (x-1) can be obtained by simply flipping all the bits to the right of rightmost 1 in x and also including the rightmost 1.

Now think about x & (x-1). x & (x-1) will have all the bits equal to the x except for the rightmost 1 in x.

Let, x = 4 = $(100)_2$

x - 1 = 3 = $(011)_2$

x & (x-1) = 4 & 3 = $(100)_2$ & $(011)_2$ = $(000)_2$

Let, x = 6 = $(110)_2$

x - 1 = 5 = $(101)_2$

x & (x-1) = 6 & 5 = $(110)_2$ & $(101)_2$ = $(100)_2$

Properties for numbers which are powers of 2, is that they have one and only one bit set in their binary representation. If the number is neither zero nor a power of two, it will have 1 in more than one place. **So if x is a power of 2 then x & (x-1) will be 0.**

Implementation: (POWER OF 2)

```
 bool isPowerOfTwo(int x)
  {
      // x will check if x == 0 and !(x & (x - 1)) will check if x is a power of 2 or
not
      return (x && !(x & (x - 1)));
  }
```

Implementation: (POWER OF 4) (***IMP******)**

```
public class Solution {
   public boolean isPowerOfFour(int num) {
     return(num > 0 && ((num & num-1) == 0) && ((Math.log(num)/
Math.log(2)) %2 == 0));
   }
}
```

$$

Count the number of ones in the binary representation of the given number.

The basic approach to evaluate the binary form of a number is to traverse on it and count the number of ones. But this approach takes $\log_2 N$ of time in every case.

Why $\log_2 N$?

As to get a number in its binary form, we have to divide it by 2, until it gets 0,

which will take $\log_2 N$ of time.

With bitwise operations, we can use an algorithm whose running time depends on the number of ones present in the binary form of the given number. This algorithm is much better, as it will reach to logN, only in its worst case.

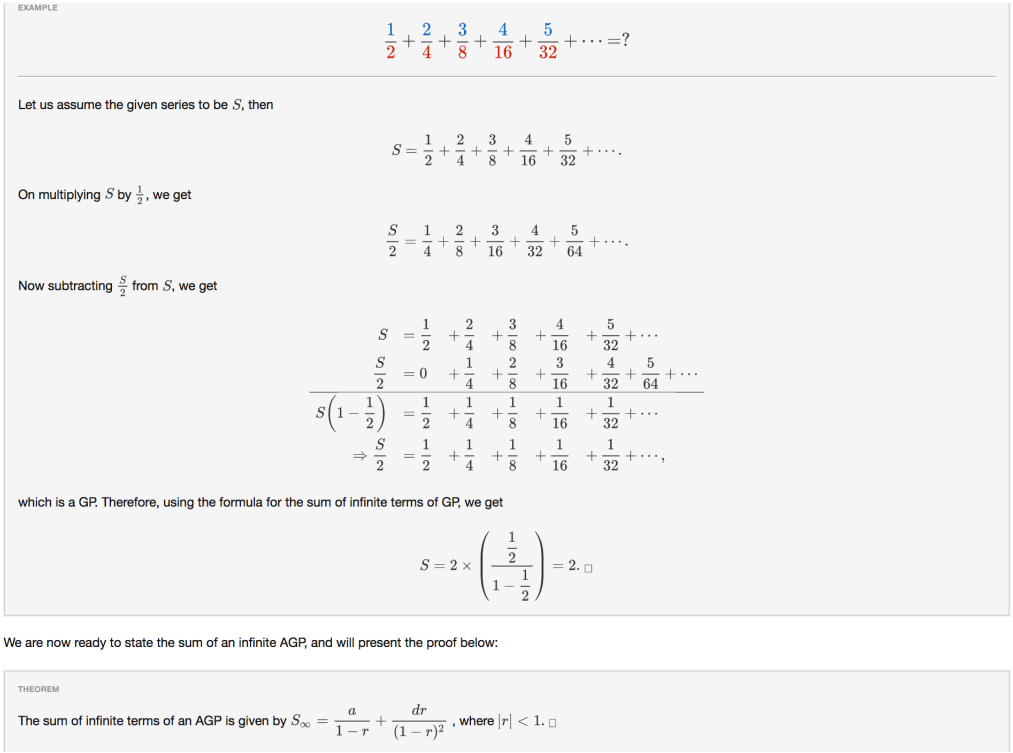
```
int count_one (int n)
   {
      while( n )
      {
      n = n&(n-1);
        count++;
      }
      return count;
}
```

$$

Pascal's triangle:

(i-1)C(j-1)

$$



EXAMPLE

$$\frac{1}{2}+\frac{2}{4}+\frac{3}{8}+\frac{4}{16}+\frac{5}{32}+\cdots=?$$

Let us assume the given series to be S, then

$$S=\frac{1}{2}+\frac{2}{4}+\frac{3}{8}+\frac{4}{16}+\frac{5}{32}+\cdots.$$

On multiplying S by $\frac{1}{2}$, we get

$$\frac{S}{2}=\frac{1}{4}+\frac{2}{8}+\frac{3}{16}+\frac{4}{32}+\frac{5}{64}+\cdots.$$

Now subtracting $\frac{S}{2}$ from S, we get

$$\begin{aligned} S &= \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} + \cdots \\ \frac{S}{2} &= 0 + \frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \frac{4}{32} + \frac{5}{64} + \cdots \\ \hline S\left(1-\frac{1}{2}\right) &= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \cdots \\ \Rightarrow \frac{S}{2} &= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \cdots, \end{aligned}$$

which is a GP. Therefore, using the formula for the sum of infinite terms of GP, we get

$$S=2\times\left(\frac{\frac{1}{2}}{1-\frac{1}{2}}\right)=2.\ \square$$

We are now ready to state the sum of an infinite AGP, and will present the proof below:

THEOREM

The sum of infinite terms of an AGP is given by $S_\infty=\frac{a}{1-r}+\frac{dr}{(1-r)^2}$, where $|r|<1.\ \square$

$$

Recursion is more time consuming and space consuming that iteration

$$

NOTE:

calloc() zero-initializes the buffer, while malloc() leaves the memory uninitialized.

$$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$$$$$

https://leetcode.com/problems/single-number-ii/description/

Read Explanation:
https://leetcode.com/problems/single-number-ii/discuss/43295/Detailed-explanation-and-generalization-of-the-bitwise-operation-method-for-single-numbers

$$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$$$$$

NUMBER OF ONES
https://leetcode.com/problems/number-of-digit-one/

For example '8192':
1-999 -> countDigitOne(999)
1000-1999 -> 1000 of 1s + countDigitOne(999)
2000-2999 -> countDigitOne(999)
.
.
7000-7999 -> countDigitOne(999)
8000-8192 -> countDigitOne(192)
Count of 1s : **countDigitOne(999)8 + 1000 + countDigitOne(192)*
Noticed that, if the target is '1192':
Count of 1s : **countDigitOne(999)1 + (1192 - 1000 + 1) + countDigitOne(192)*
(1192 - 1000 + 1) is the 1s in thousands from 1000 to 1192.
Same codes as above, maybe much easier to understand.

/*
233. Number of Digit One

Given an integer n, count the total number of digit 1 appearing in all non-negative integers less than or equal to n.

For example:
Given n = 13,
Return 6, because digit 1 occurred in the following numbers: 1, 10, 11, 12, 13.

```
*/
class Solution {
public:
   int lengthDigit(int n){
      int c=0;
      while(n>0){
         c++;
         n=n/10;
      }
      return c;
   }
   int countDigitOne(int n) {
      if(n<=0){
         return 0;
      }else if(n<10){
         return 1;
      }
      int length=lengthDigit(n);
      int base=pow(10,length-1);
      int answer=n/base;
      int remainder=n%base;
      int m;
      if(answer==1){
         m=n-base+1;
      }else{
         m=base;
      }
      return countDigitOne(base-1)*answer+m+countDigitOne(remainder);
   }
};
```

$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$. $$$$$$$$

MAJORITY VOTING ALGORITHM

https://gregable.com/2013/10/majority-vote-algorithm-find-majority.html

$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$. $$$$$$$$

//remember if the numbers are too large the to find mid dont add up the two nos
//just do s+(e-s)/2 and then find mid also avoid recussion in such cases.

$$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$. $$$$$$$$

Tortroise and hare algorithm to find duplicates in array and also cycle in list or same numbers.
****Linklist and array duplicates both**
http://codingfreak.blogspot.com/2012/09/detecting-loop-in-singly-linked-list_22.html

$$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$. $$$$$$$$

Population of boys and girl in country:
https://www.geeksforgeeks.org/puzzle-17-ratio-of-boys-and-girls-in-a-country-where-people-want-only-boys/

$$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Binary Indexed Tree (for long complexity which in <n for sum and update values in array)

We have to use binary indexed. tree for such questions:
https://www.geeksforgeeks.org/binary-indexed-tree-or-fenwick-tree-2/

getSum(index): Returns sum of arr[0..index]
// Returns sum of arr[0..index] using BITree[0..n]. It assumes that
// BITree[] is constructed for given array arr[0..n-1]
1) Initialize sum as 0 and index as index+1.
2) Do following while index is greater than 0.
...a) Add BITree[index] to sum
...b) Go to parent of BITree[index]. Parent can be obtained by removing
the last set bit from index, i.e., index = index - (index & (-index))
3) Return sum.

update(index, val): Updates BIT for operation arr[index] += val
// Note that arr[] is not changed here. It changes
// only BI Tree for the already made change in arr[].
1) Initialize index as index+1.

2) Do following while index is smaller than or equal to n.
...a) Add value to BITree[index]
...b) Go to parent of BITree[index]. Parent can be obtained by removing
the last set bit from index, i.e., index = index + (index & (-index))

$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

TWO STRING HAVE COMMON CHARACTERS IN. BETWEEN OR NOT

MAKE STRING TO VALUE AND IF AND OF TWO VALES IS NOT 0 THEN THEY HAVE COMNON CHAR

MAKING VAUES:
for(int i=0;i<w.length();I++){
Int k = (w[I]-'a');
val=val | (1<<k);
}

Similarly for second string too.

And if val1 & val2 ==0 then no char in common.
$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

MULTISET in C++ is already sorted

std::**multiset**. std::**multiset** is an associative container that contains a **sorted** set of objects of type Key. Unlike set, multiple keys with equivalent values are allowed.**Sorting** is done using the key comparison function

$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Stability of sorting algorithm

A sorting algorithm is said to be **stable** if two objects with equal keys appear in the same order in sorted output as they appear in the input array to be sorted. Some sorting algorithms are stable by nature like Insertion sort, Merge Sort, Bubble Sort, etc. And some sorting algorithms are not, like Heap Sort, Quick Sort, etc.
Background: a "stable" sorting algorithm keeps the items with the same sorting key in order. Suppose we have a list of 5-letter words:
peach
straw
apple

spork
If we sort the list by just the first letter of each word then a stable-sort would produce:
apple
peach
straw
spork

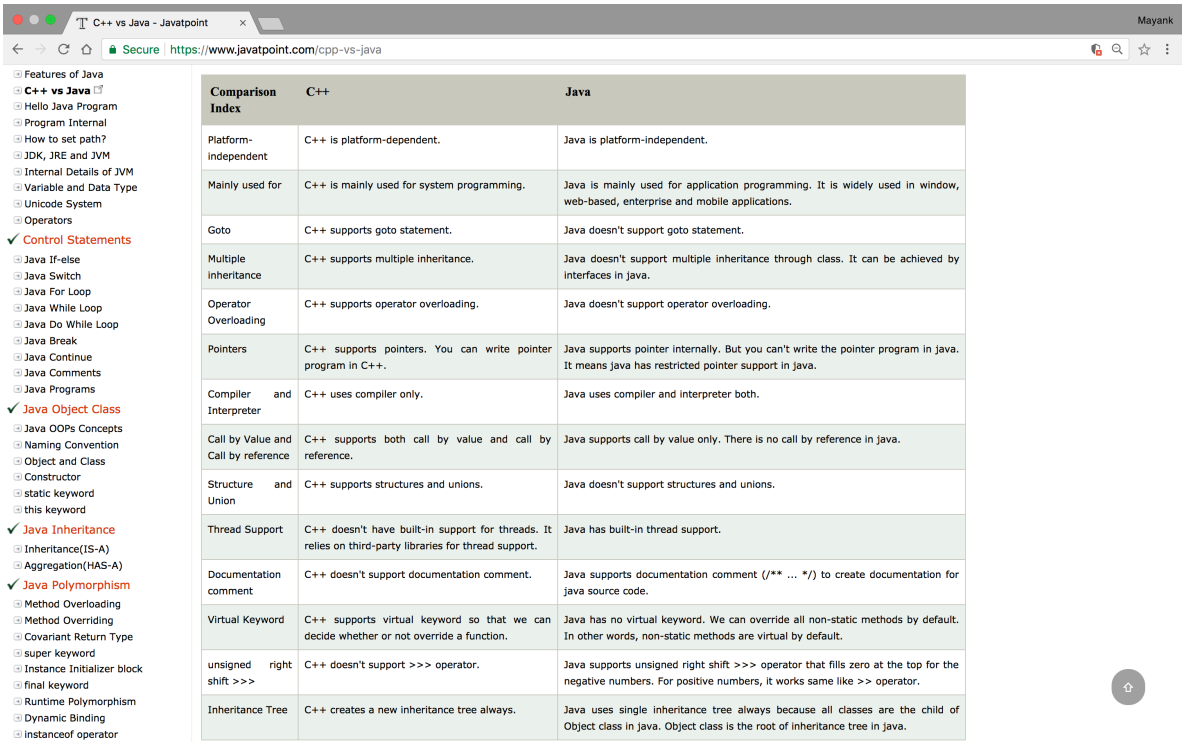
$$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

unget(int c,File *fp)

It put back the character reducing the file pointer by one as if the previous gets operation is undone.

$$$$$$$$$$$$$$$$$$$$. VERY IMPORTANT $$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Number Theory
1. Modular Exponentiation
2. Modular multiplicative inverse
3. Primality Test | Set 2 (Fermat Method)
4. Euler's Totient Function
5. Sieve of Eratosthenes
6. Convex Hull
7. Basic and Extended Euclidean algorithms
8. Segmented Sieve
9. Chinese remainder theorem
10. Lucas Theorem



Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Goto	C++ supports goto statement.	Java doesn't support goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. But you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only.	Java uses compiler and interpreter both.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses single inheritance tree always because all classes are the child of Object class in java. Object class is the root of inheritance tree in java.

Sort map by value

https://www.geeksforgeeks.org/sort-elements-frequency-set-4-efficient-approach-using-hash/

Get line c++

getline(cin,s);