Aim: - To solve Modular Linear Equations .

Inputs: - ***N***, ***a*** ,***b***.

Output :- x i.e. solution to the equation in .

Solution:-

Q2a🡪 Now for solution to this question we are given the following data as follows: -

N= 64380800680355443923012985496149269915138610753401343291807343952413826484237 0630061369715394739134090922937332590384720397133335969549256322620979036686633213903952966175107096769180017646161851573147596390153.

a= 34325464564574564564768795534569998743457687643234566579654234676796634378768 434237897634345765879087764242354365767869780876543424.

b= 45292384209127917243621242398573220935835723464332452353464376432246757234546 765745246457656354765878442547568543334677652352657235.

Now to proceed further we will declare 3 BigIntegers in Java using BigInteger Class. Now let’s have a look at the equation.

Now subtracting ” ***b*”**  from both sides, we get the following equation: -

.

Now multiplying both sides by . (As gives us 1).

Now we are left with the following equation: -

Now to proceed further we need to calculate the for the equation. First we must check whether exists or not. Now we can perform that check by calculating the GCD(a,N). if the GCD(a,N) is 1, it means the inverse exists. Also a simple check should can also be performed before calculating the GCD, i.e. if N and a are even we should terminate the test there itself as GCD will be greater than 1 in that case. (We can perform odd and even test by just looking at last digit.)

To find we have used library method of BigInteger Class. And by performing the calculations we get the following result of x as follows: -

Value of x before modulus

.

Value of x after Modulus

x=421183184045396008847949672535189703891596775623386918532288953516218079938085028752886155666620391419813631317167316631819337277694338966865265179264143550762845428781671313603290317056892475467340480506865970

The Value has been tested in the program by replacing x with the value above.

Q2b --> Given the following are the values for equation:-

N=34248723532593458235023583785345602939423526832829428589598243238758257023423 8487625923289526382379523573265963293293839298595072093573204293092705623485273893582930285732889238492377364284728834632342522323422

a=34325464564574564564768795534569998743457687643234566579654234676796634378768 434237897634345765879087764242354365767869780876543424

b=24243252873562935279236582385723952735639239823957923562835832582635283562852 252525256882909285959238420940257295265329820035324646

Now in the same way we will proceed further to calculate the inverse. But before that we will perform our preliminary test.

From our observation we can see that the number a is not invertible as both Numbers N and a are Even. So we can say that the GCD(a,N) will not be equal to 1. So there is no point in going further. And we can directly say the there is no solution to the equation with above values.

**Question 3**

From the observations it is seen that in 2a N was odd and a was even. So we calculated the GCD i.e. 1. Now a number to have its inverse in Zn, the GCD(x,N) (x refer to the number) should be equal to 1. This is a pre requisite for having a inverse. If this condition is not satisfied, we cannot calculate the inverse of a number. Even Java library function throws an Exception when such numbers are provided whose GCD is not equal to 1.

Whereas in 2b N and a both were even. And GCD of two even numbers cannot be 1. It will be greater than 1. Hence we cannot find the inverse of a. Therefore there is no solution to the equation. And also in 2a N is the Modulus of the group.

Now next comes the comparison of running times for calculating the inverse.

Fermat’s little theorem states that if m is a prime and a is an integer co-prime to m, then ap − 1 will be evenly divisible by m. That is or

And the running time of Fermat’s theorem will be approximately roughly i.e. Cubic. As it is using values that are exponential, so calculations will higher. n stands for the number of bits in a number. And since we have to deal with very large numbers such as Big Integers we can’t afford to use Fermat’s Theorem. Whereas in the Library method it is very efficient and has a constant time of running independent of number of bits.