Basic\_of\_c\_2

Bitwise not operator(~) - it is also a unary operator (i.e those operator which need only operant to perform their work)

As in the 32 or 64 bits -

Integer value will take 4 bytes - i.e 32 bits

Thus

Binary of 5 is 00000000000000000000000000000101

i.e (<0\*29>101) which we normally represented by 101

And by (~) bitwise not operator all the zeros are converted to 1 and all the 1 are converted to 0

I.e (~5 = 11111111111111111111111111111010)

So tabhi only (101) galat hai kyu ki us me bahot sare zero hote h jin ko 1 banana hai bitwise not operator me

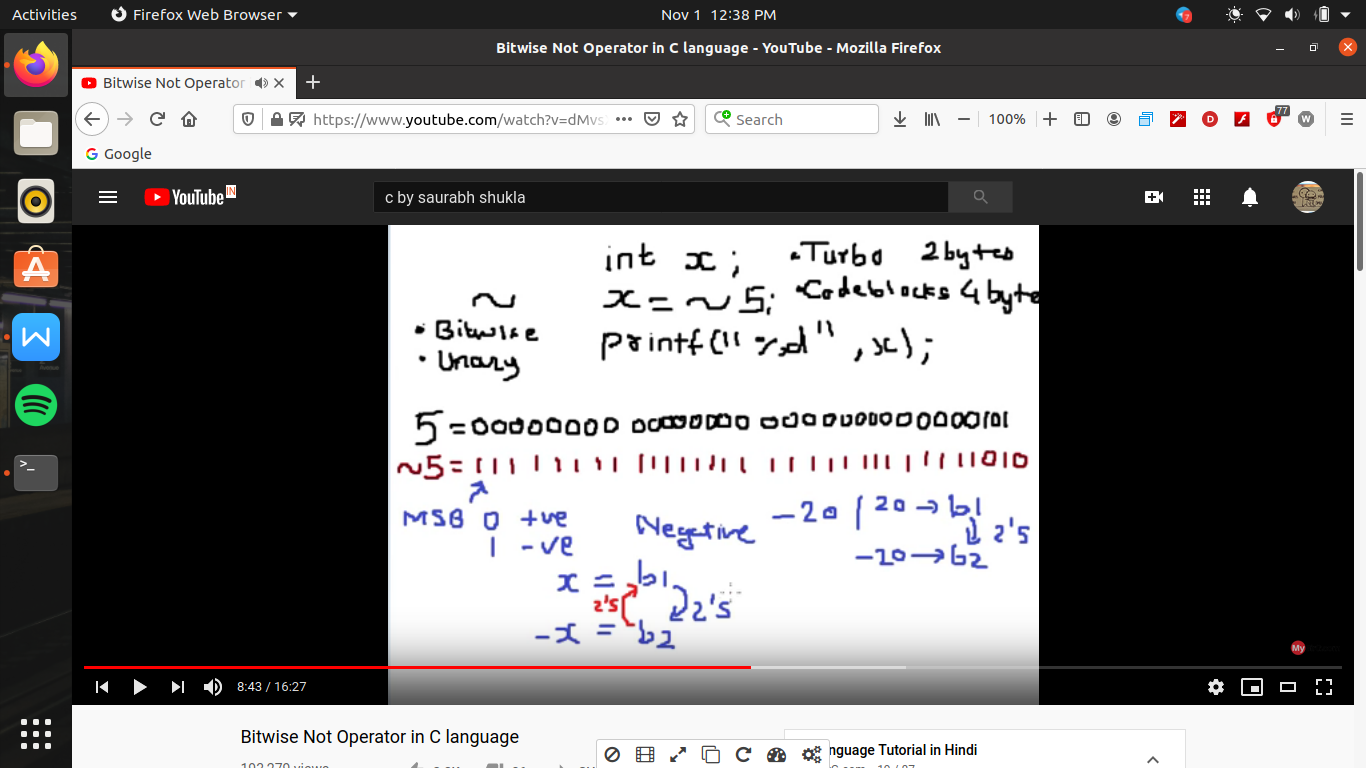
Here jo

Pahali wali bit hoti hai use kahate h most significant bit

And if first bit is 0 then it means no. Is positive(+)

And if 1 -> (-ve)

So bitwise not operator ke use karane se +ve value negative ho gayi



Type of bitwise operator ->

1. Bitwise AND &
2. Bitwise OR |
3. Bitwise XOR ^ // i.e excluseive or
4. Bitwise NOT ~
5. Right shift >>
6. Left shift <<

As bitwise operator are work in a binary no’s

So those operants are 0 and 1

&operator ->

And operations is works as same as and gate

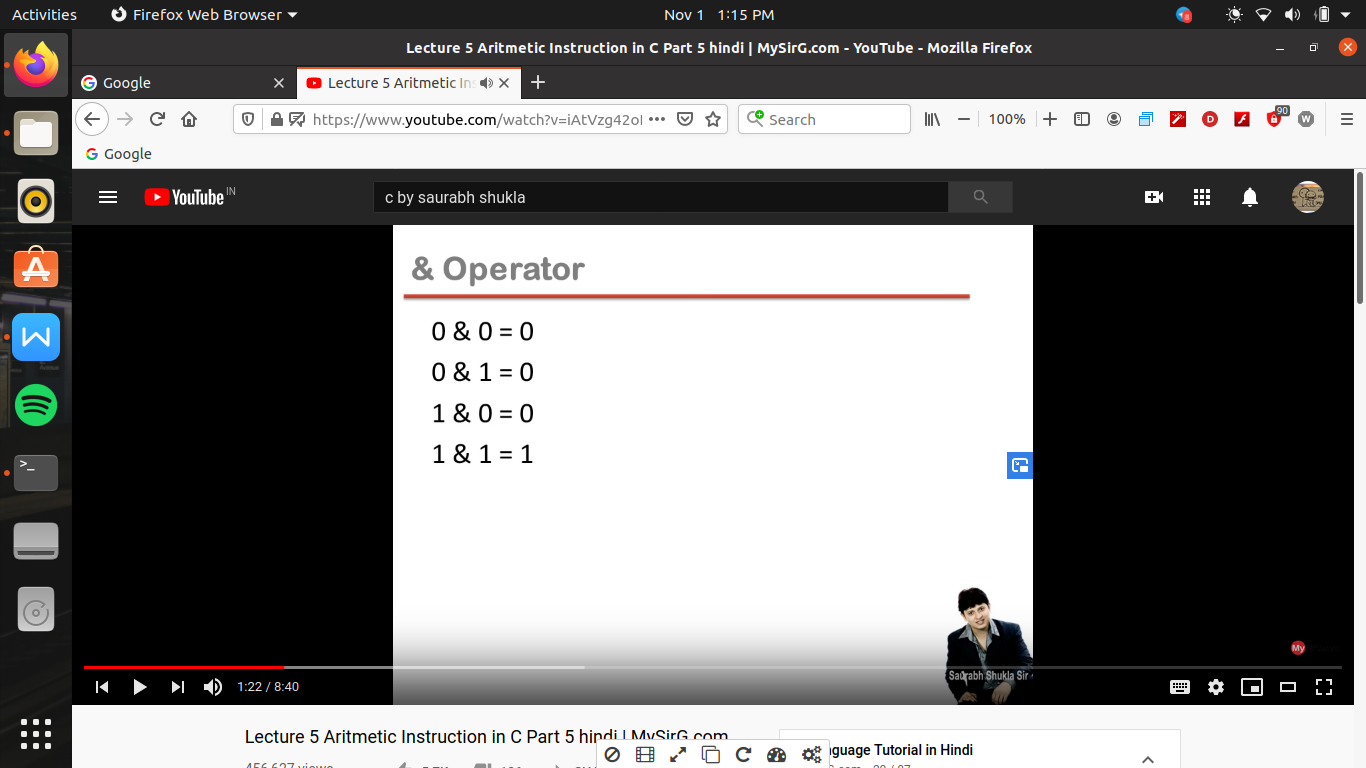
1. e agar dono me se koe 0 hoga toh us ka result bhi 0 ayega)

0&0=0

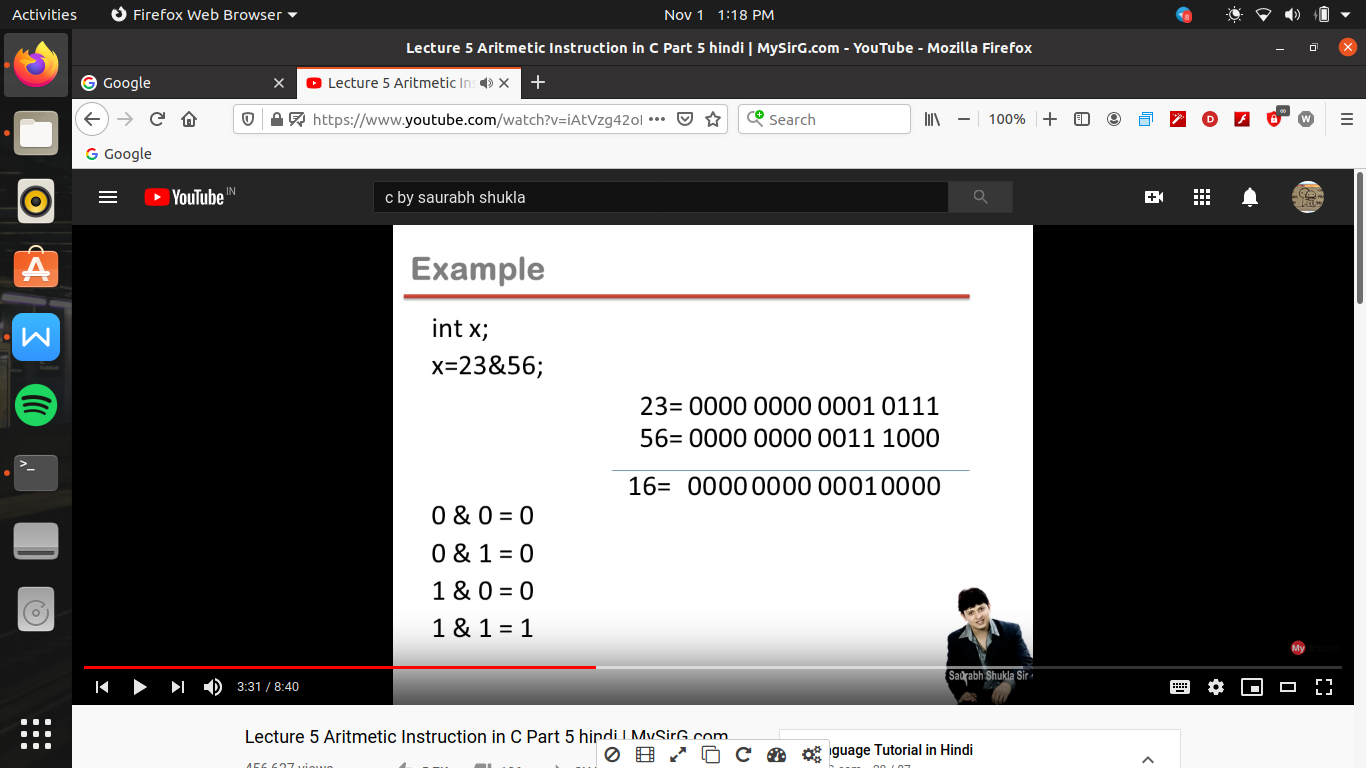
0&1= 0

1&0 =0

1&1 =1



Ex-



OR (|) operator ->

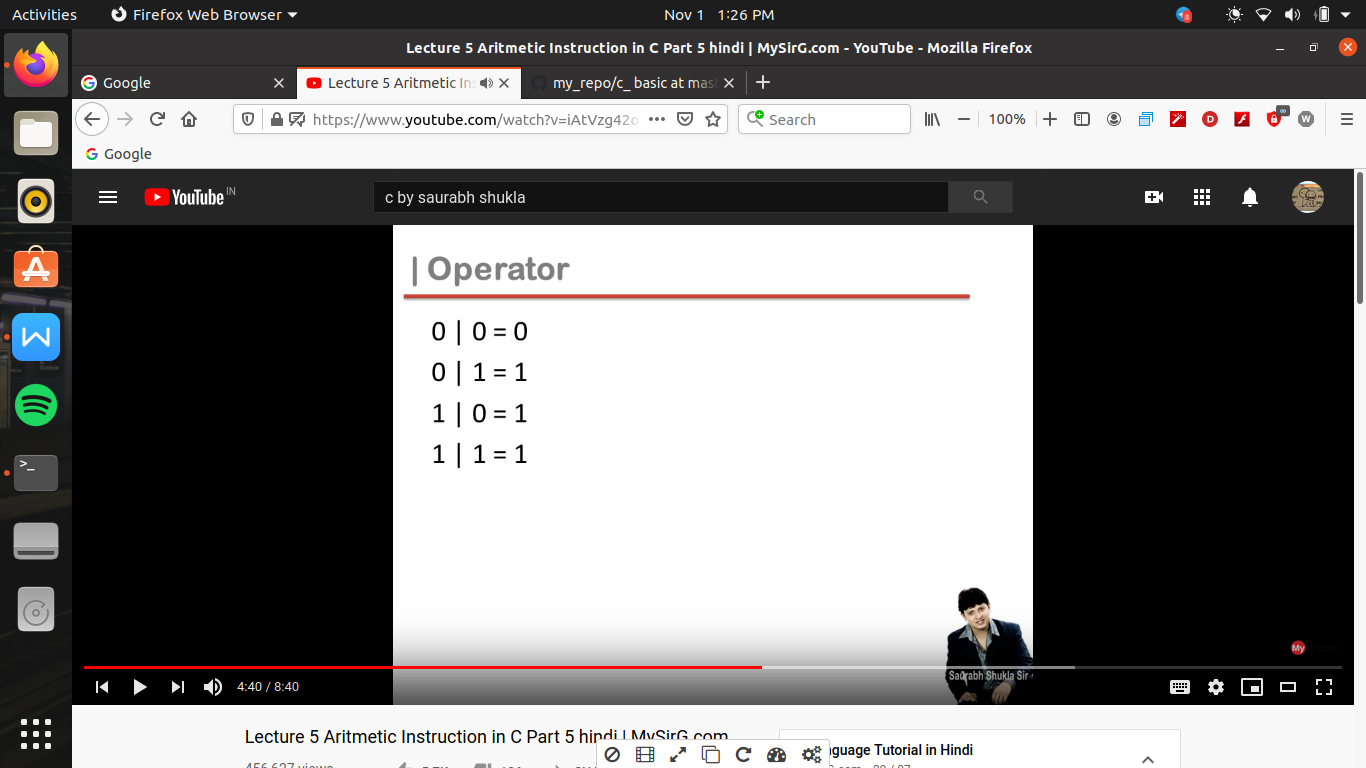
Es me if dono hi 0 hai tabhi result 0 ayega nhi toh 1

0|0=0

0|1=1

1|0=1

1|1=1



XOR operator (^) ->

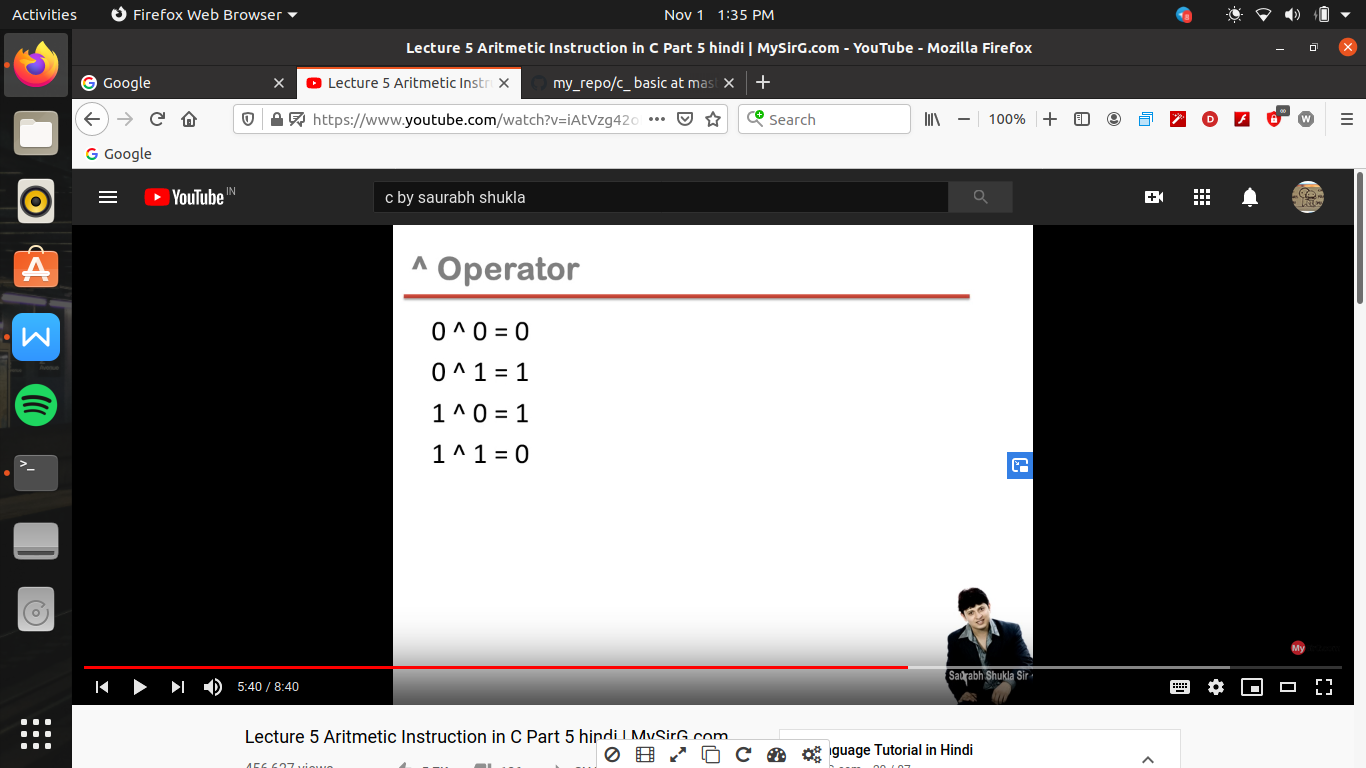
Es me dono operants same hai toh result 0 ayega or agar dono operants alag hai toh 1 ayega

0^0=0

0^1=1

1^0=1

1^1=0

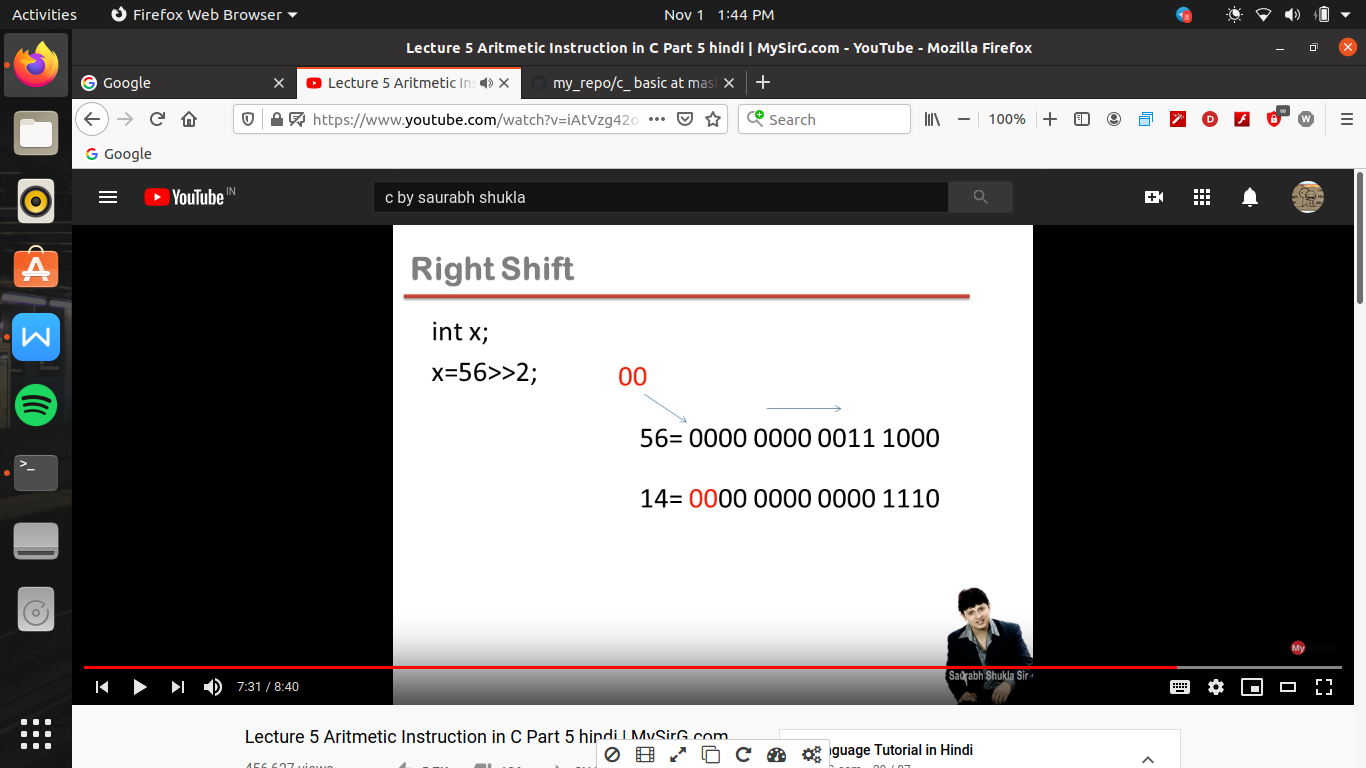


Right shift operation (>>)->

It is a operator in the we input a constant of which we have to right shift of the variable’s binary value

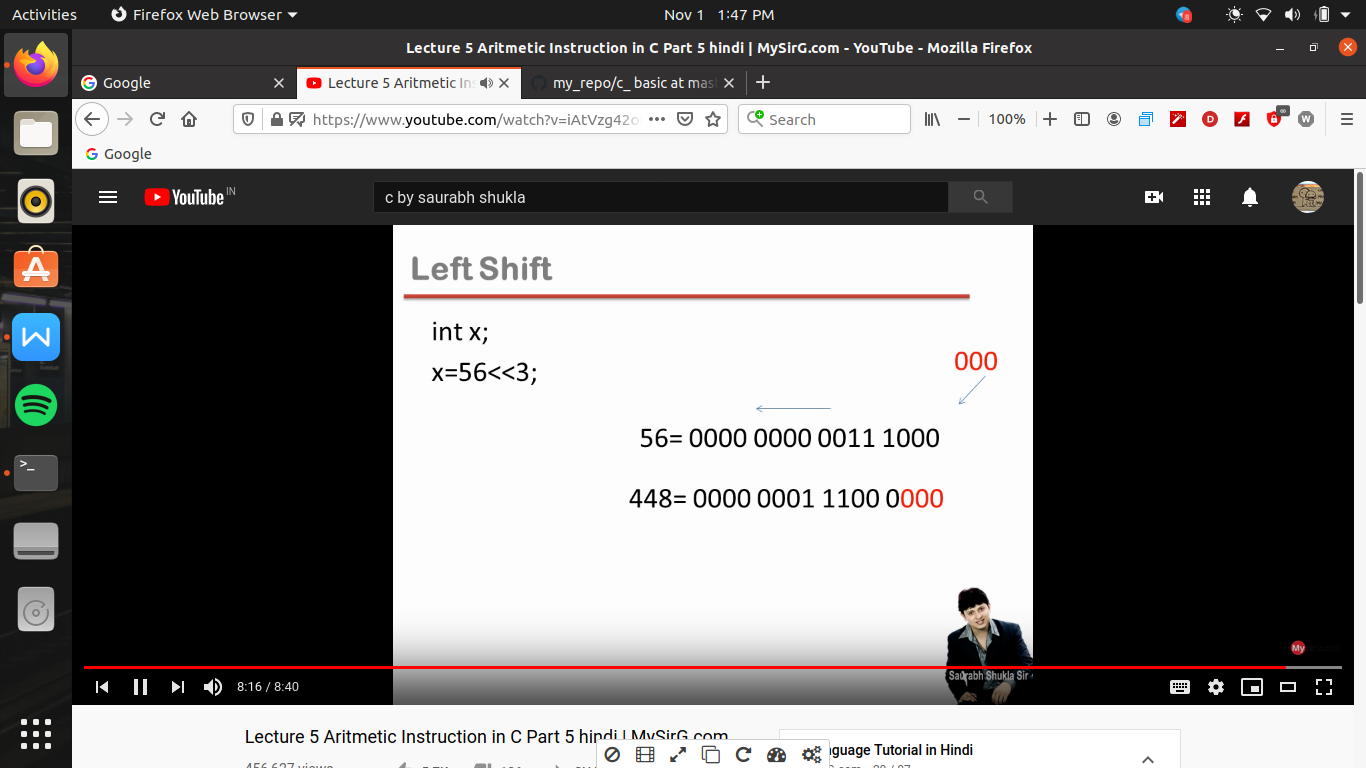
Ex

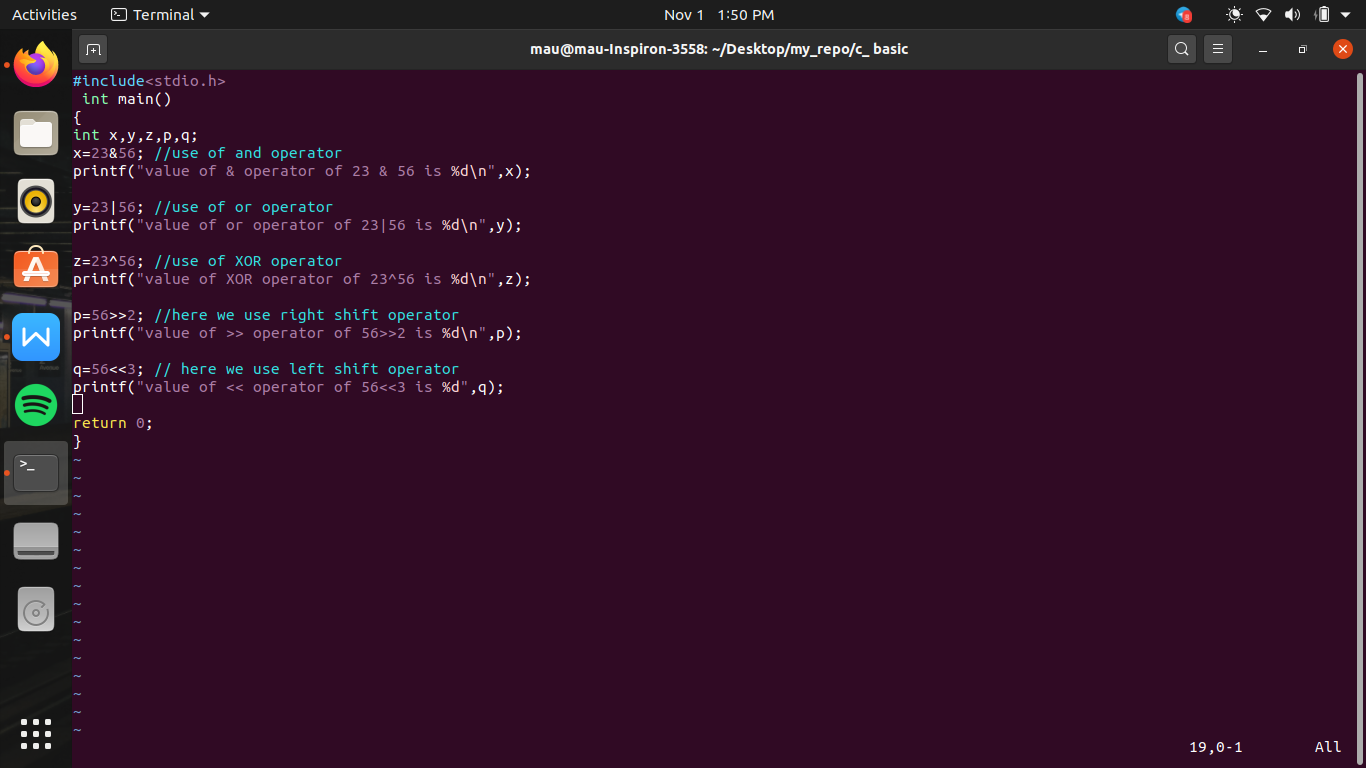
56>>2 toh 56 ke binary nos me hum 2 zero left size enter karege or baki value right ki or shift ho jayegi , or yaha size of a bit is same toh last ki 2 value delete ho jayegi

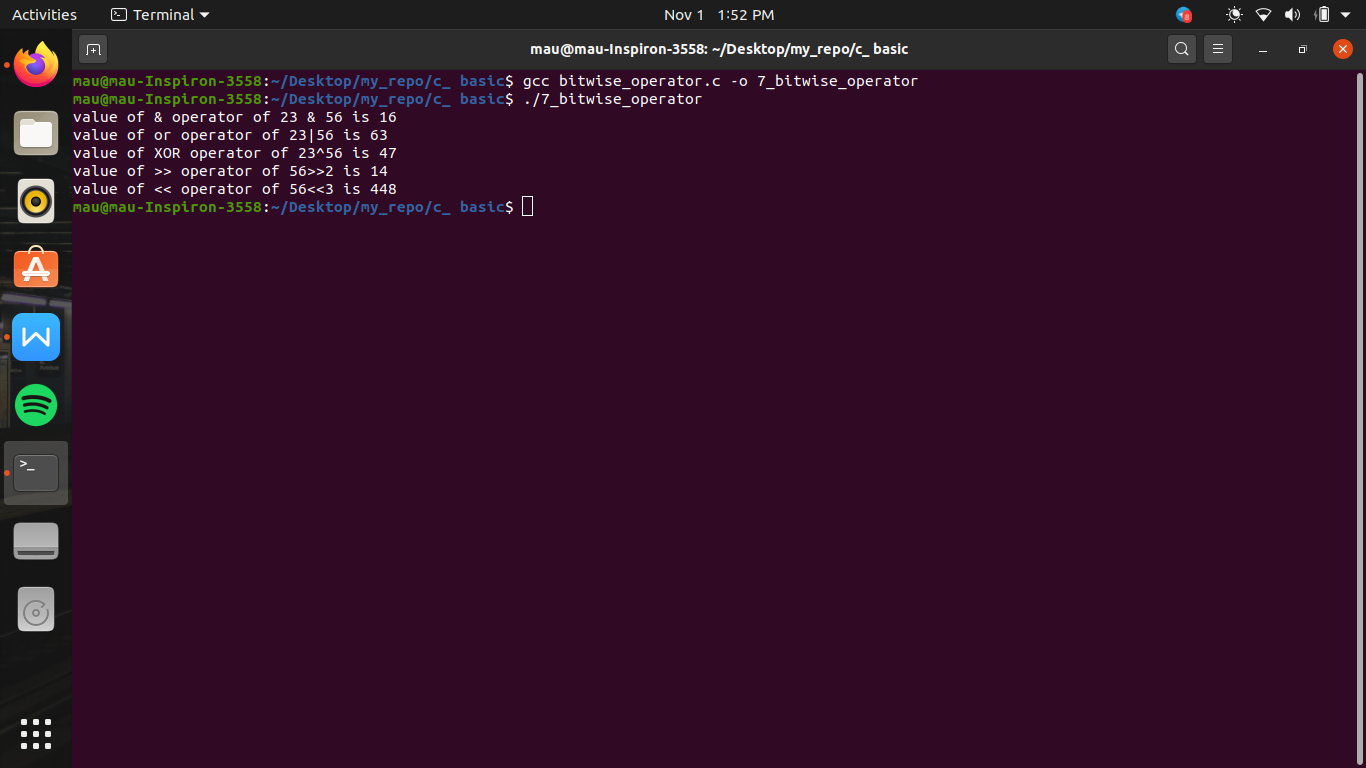


Left shift (<<) operator->

Now here hum zeros ko right side se enter karege or extreme left side wale binary no. Delete ho jayege







Relational operators->

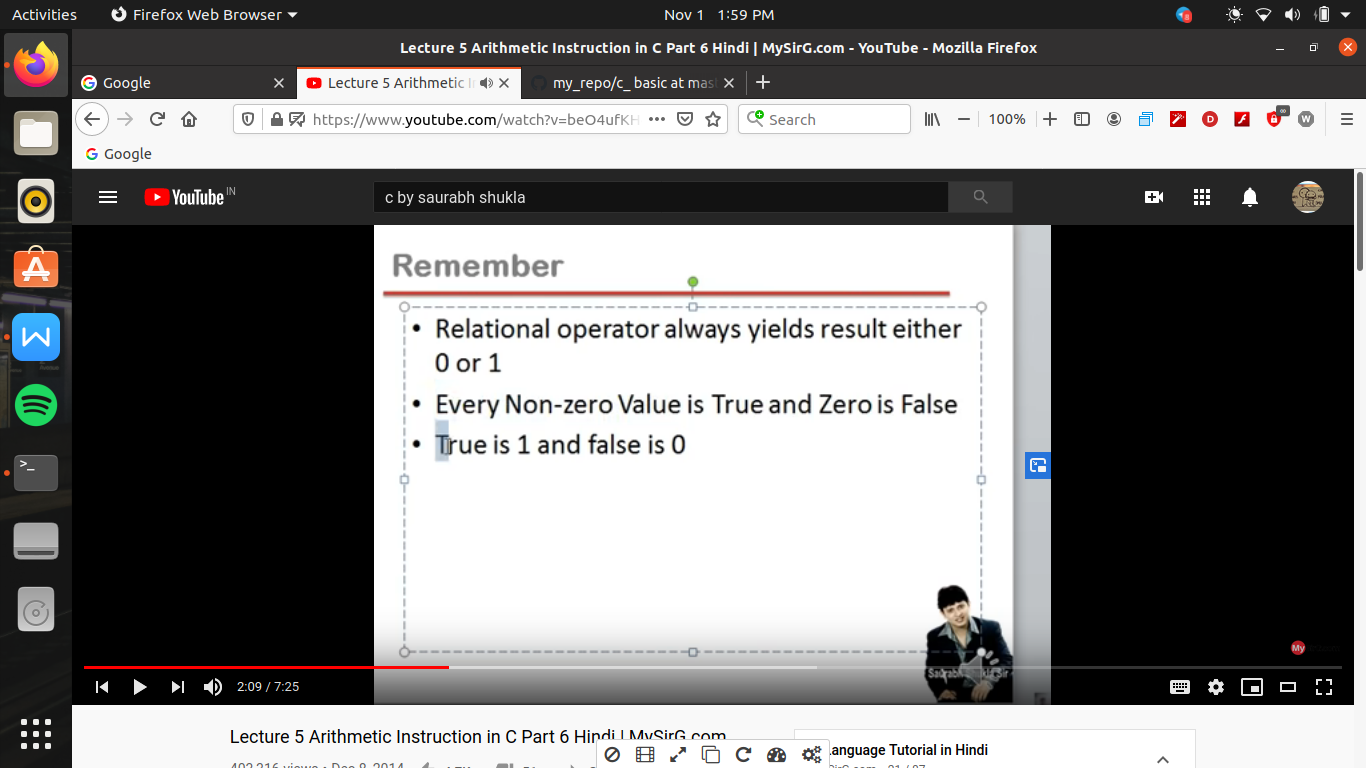
These are of 6 types

< , > ,<= , >= (en ki priority neeche likhe operators se jayada hai )

==(equal to) , !=(not equalt to)

Rules of relational operator ->

1. Relational operator ka result hamesha ya toh 0 ata hai or ya hi 1
2. Har non 0 wali true mani jati h balaki 0 value ko faulse manate hai
3. And true means 1 and false means 0



#include<stdio.h>

int main()

{

int x,y,z,p,q;

x=3>4; //here first run relational operator i.e > then run assignment operator , so here statement is false thus the value will be 0

printf("value of x=3>4 is %d\n",x);

y=3<=4; // here statement is true thus answer will come 1

printf("value of y= 3<=4 is %d\n",y);

z=4!=3; // 4 is not equal to 3 which is true thus result yield 1

printf("value of z= 4!=3 is %d\n",z);

p=5>4>3; // here their is two same priority operator thus , operator run from left so first run 5>4 i.e true so es ka result 1 ho gaya now second operator will run i.e 1>3 // 1 because 5>4 true and its value gets replaced by 1 so as a whole result yield 0

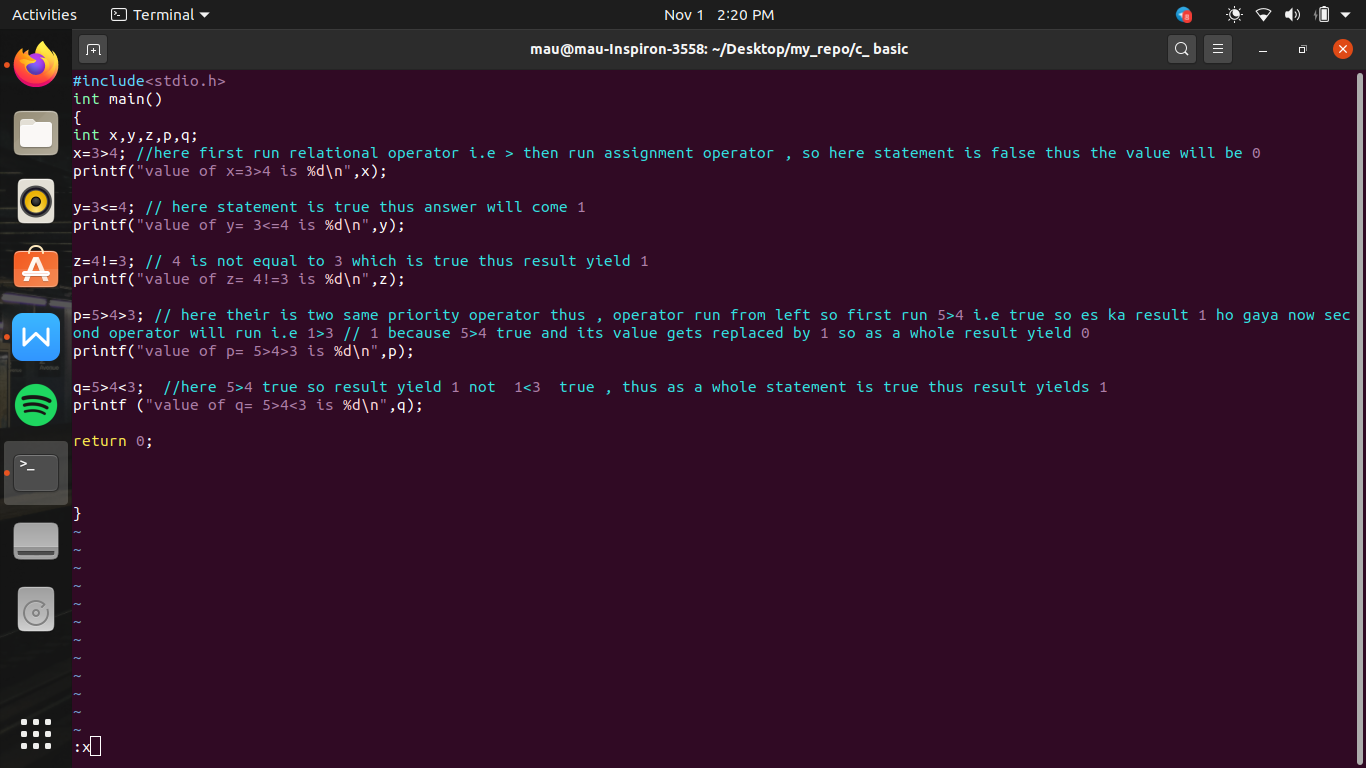
printf("value of p= 5>4>3 is %d\n",p);

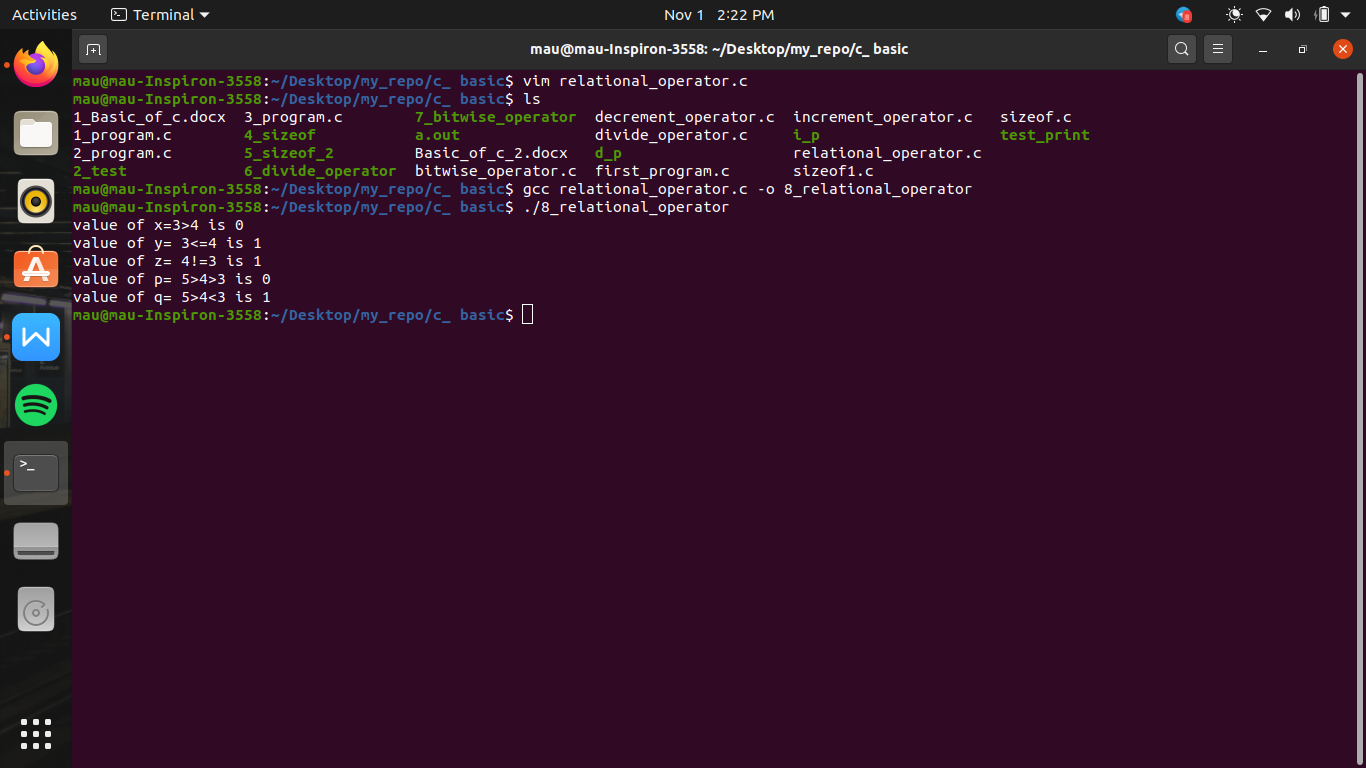
q=5>4<3; //here 5>4 true so result yield 1 not 1<3 true , thus as a whole statement is true thus result yields 1

printf ("value of q= 5>4<3 is %d\n",q);

return 0;

}





Logical operators ->

1. NOT (!)
2. AND (&&)
3. OR (||)

These are written in the sequence on the basis of their priority

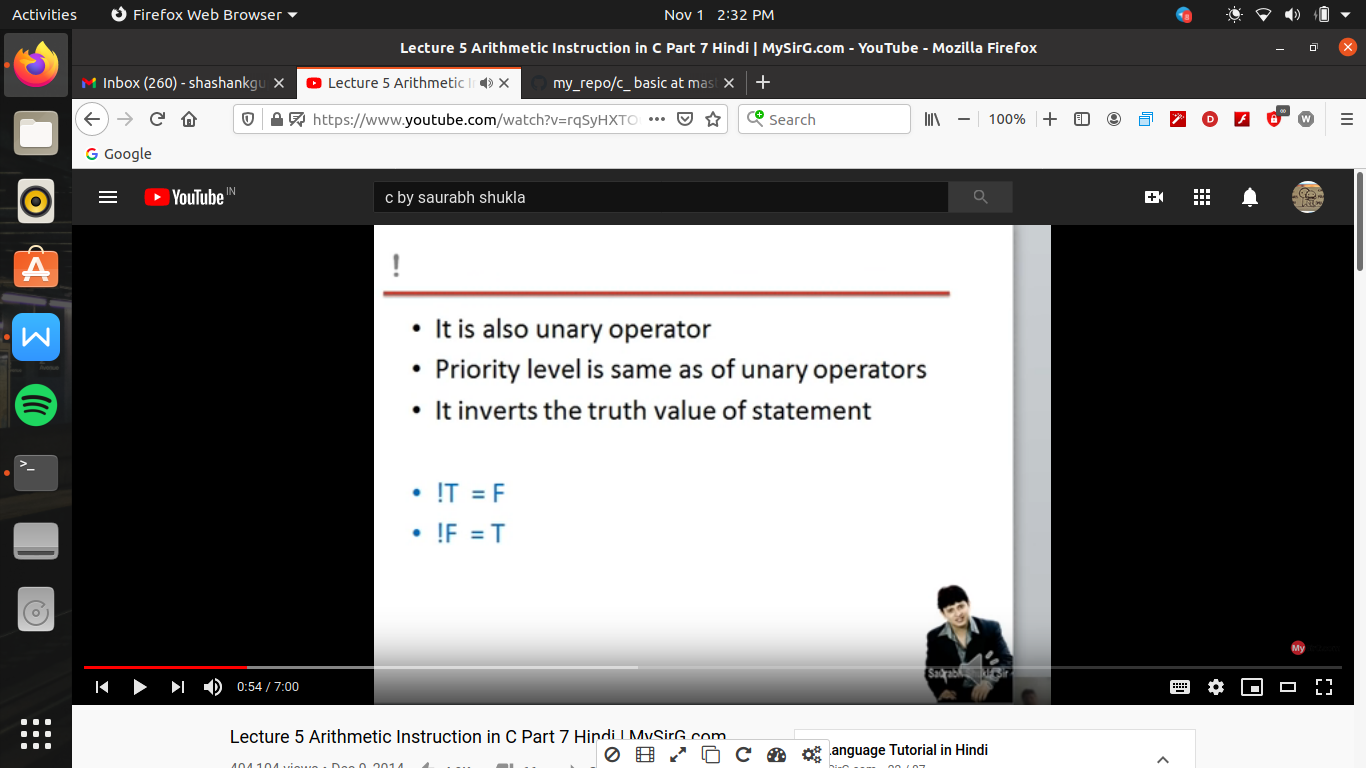
Not operator (!)->

! operator is also a unary operator

And its priority bhi same hoti hai unary operators ki tarah

Yeh statement ki true value ko badal dega

Yani true hai toh false kar dega or false hai toh true kar dega



And operator (&&)->

It is a binary operator (i.e us ko perform karane ke leye 2 operant ki jarurat hoti hai )

In and operator 2 statement ko jodane ka kam karata hai or es me hamesha pahala statement hi run karega i.e left wala chahe jo bhi expression hai

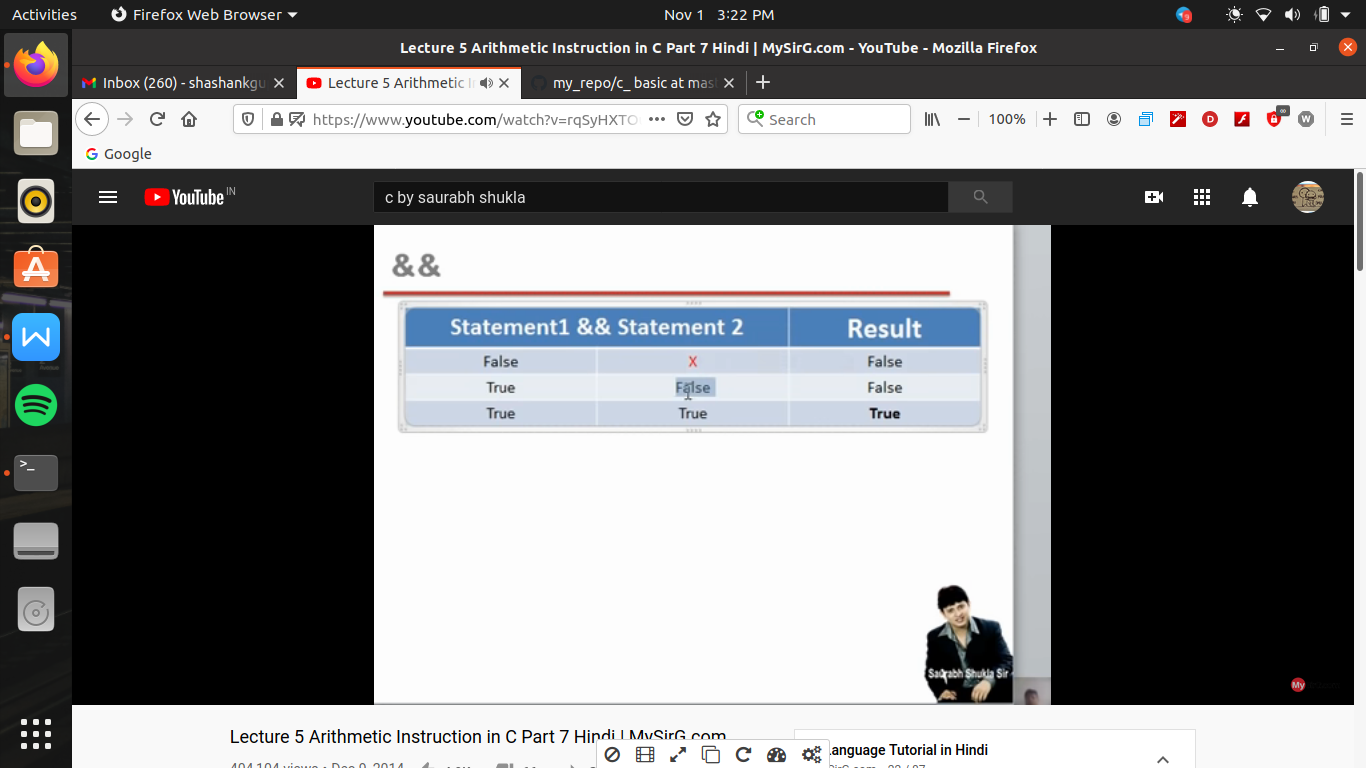
Or agar pahala wala false hai toh dusara wala deekhane ki bhi jarurat nhi hai pura expression hi false hoga

And if first statement is true the we check the second statement and if its yield false then the whole expression become false

And if 1 statement - true

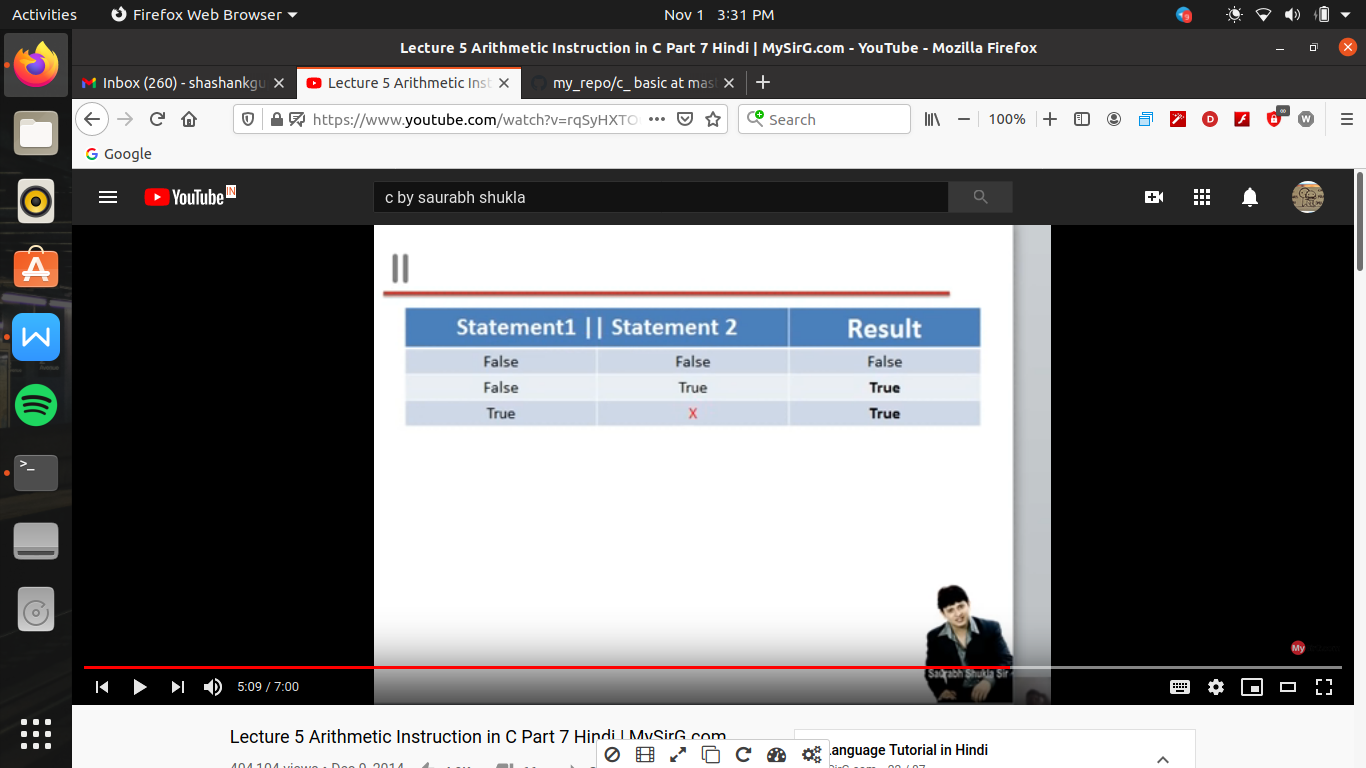
2 statement - true then only whole expression is true

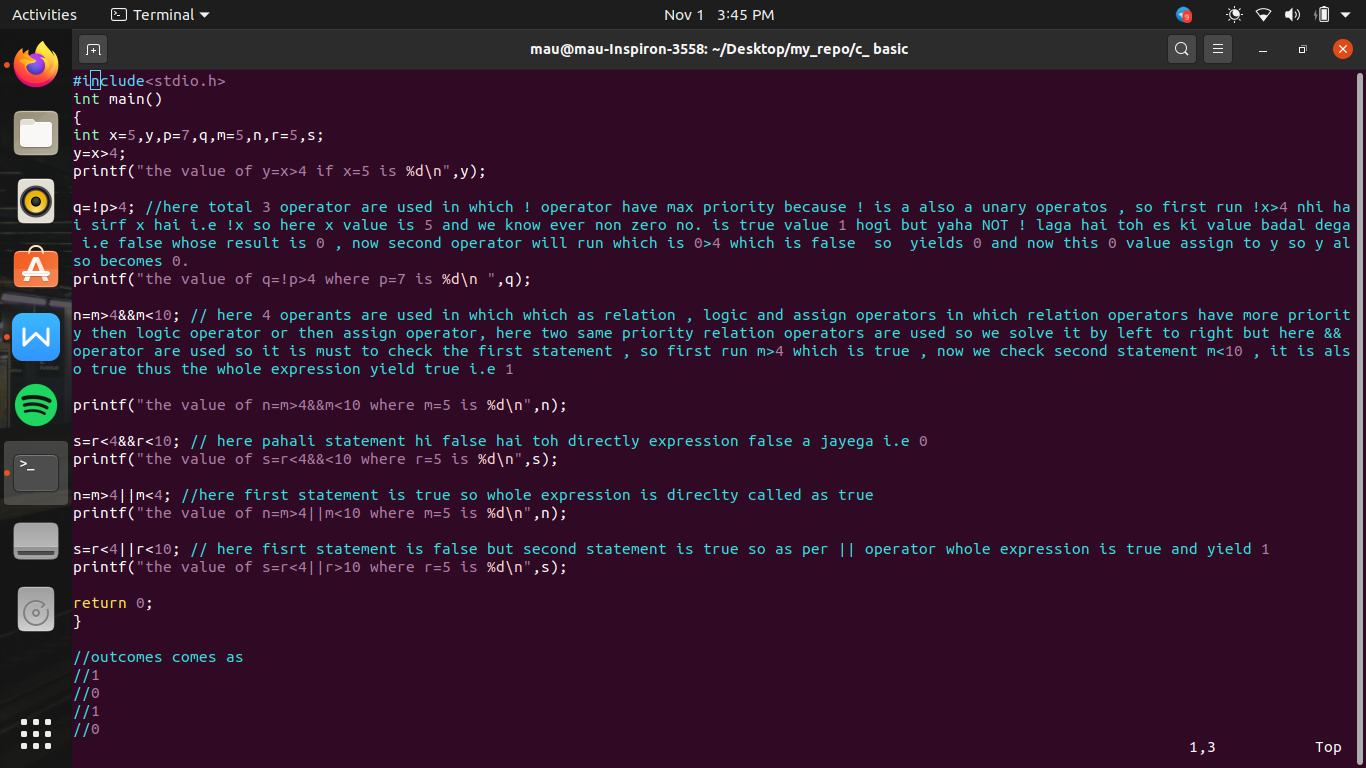
Note -> for the whole expression is true it is necessary that all the statement must be true

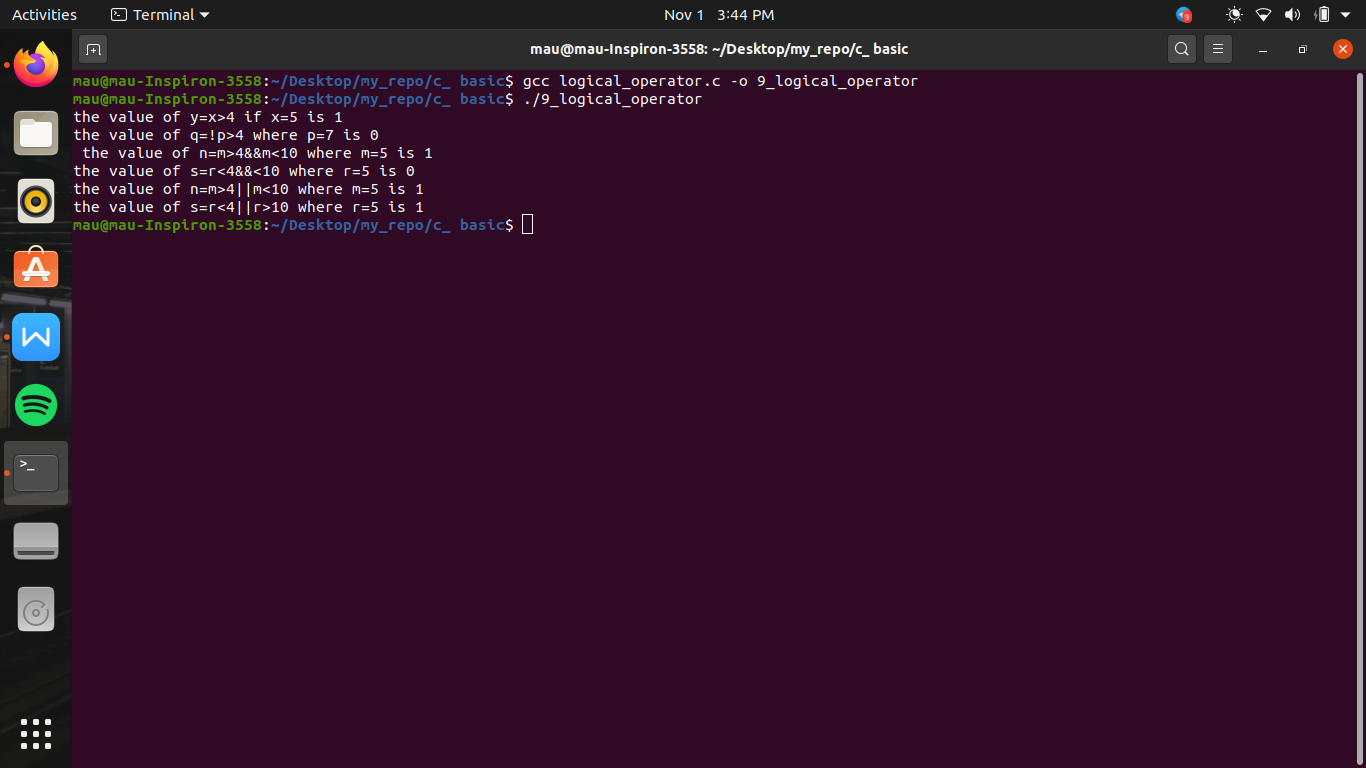


OR operator (||) ->

Or me agar dono me ek bhi statement true hai toh true ayega i.e agar dono false hai tabhi result false hoga







#include<stdio.h>

int main()

{

int x=5,y,p=7,q,m=5,n,r=5,s;

y=x>4;

printf("the value of y=x>4 if x=5 is %d\n",y);

q=!p>4; //here total 3 operator are used in which ! operator have max priority because ! is a also a unary operatos , so first run !x>4 nhi hai sirf x hai i.e !x so here x value is 5 and we know ever non zero no. is true value 1 hogi but yaha NOT ! laga hai toh es ki value badal dega i.e false whose result is 0 , now second operator will run which is 0>4 which is false so yields 0 and now this 0 value assign to y so y also becomes 0.

printf("the value of q=!p>4 where p=7 is %d\n ",q);

n=m>4&&m<10; // here 4 operants are used in which which as relation , logic and assign operators in which relation operators have more priority then logic operator or then assign operator, here two same priority relation operators are used so we solve it by left to right but here && operator are used so it is must to check the first statement , so first run m>4 which is true , now we check second statement m<10 , it is also true thus the whole expression yield true i.e 1

printf("the value of n=m>4&&m<10 where m=5 is %d\n",n);

s=r<4&&r<10; // here pahali statement hi false hai toh directly expression false a jayega i.e 0

printf("the value of s=r<4&&<10 where r=5 is %d\n",s);

n=m>4||m<4; //here first statement is true so whole expression is direclty called as true

printf("the value of n=m>4||m<10 where m=5 is %d\n",n);

s=r<4||r<10; // here fisrt statement is false but second statement is true so as per || operator whole expression is true and yield 1

printf("the value of s=r<4||r>10 where r=5 is %d\n",s);

return 0;

}

//outcomes comes as

//1

//0

//1

//0

//1

//1