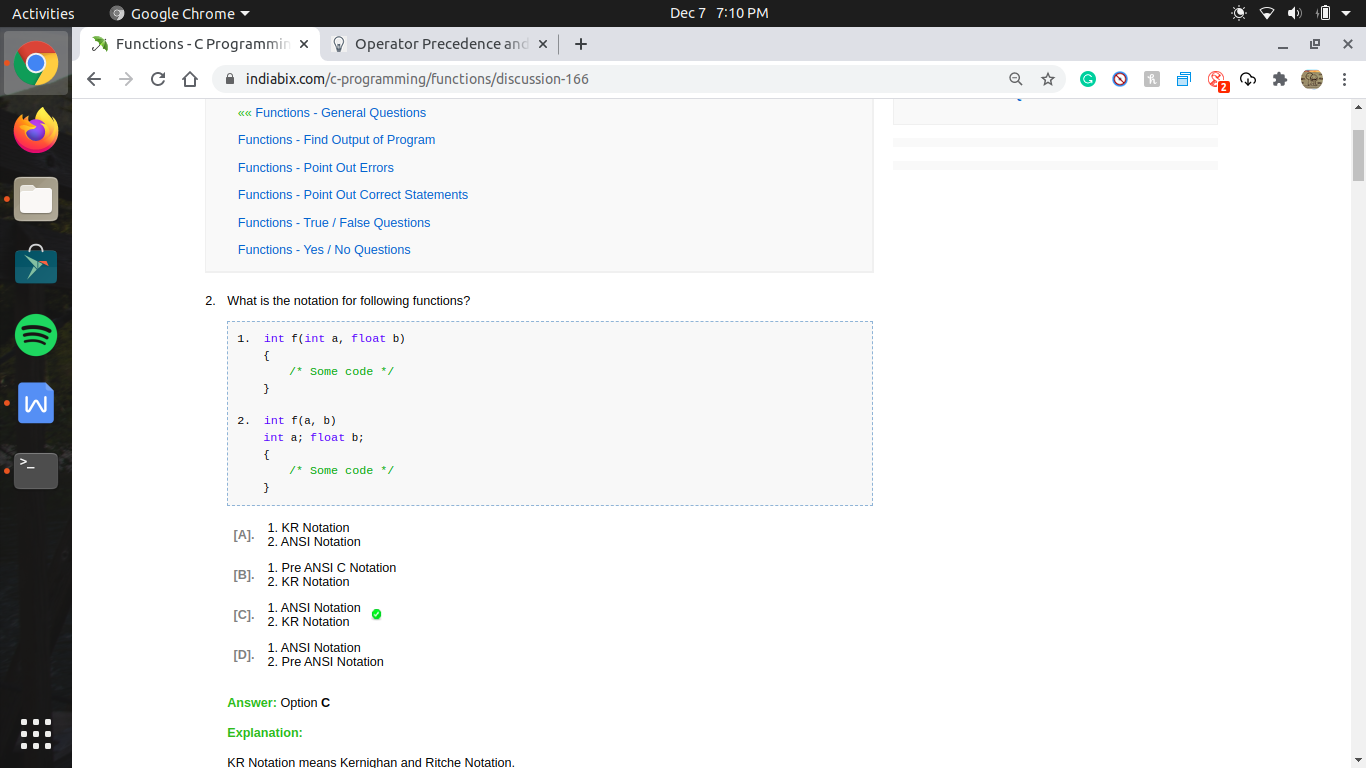
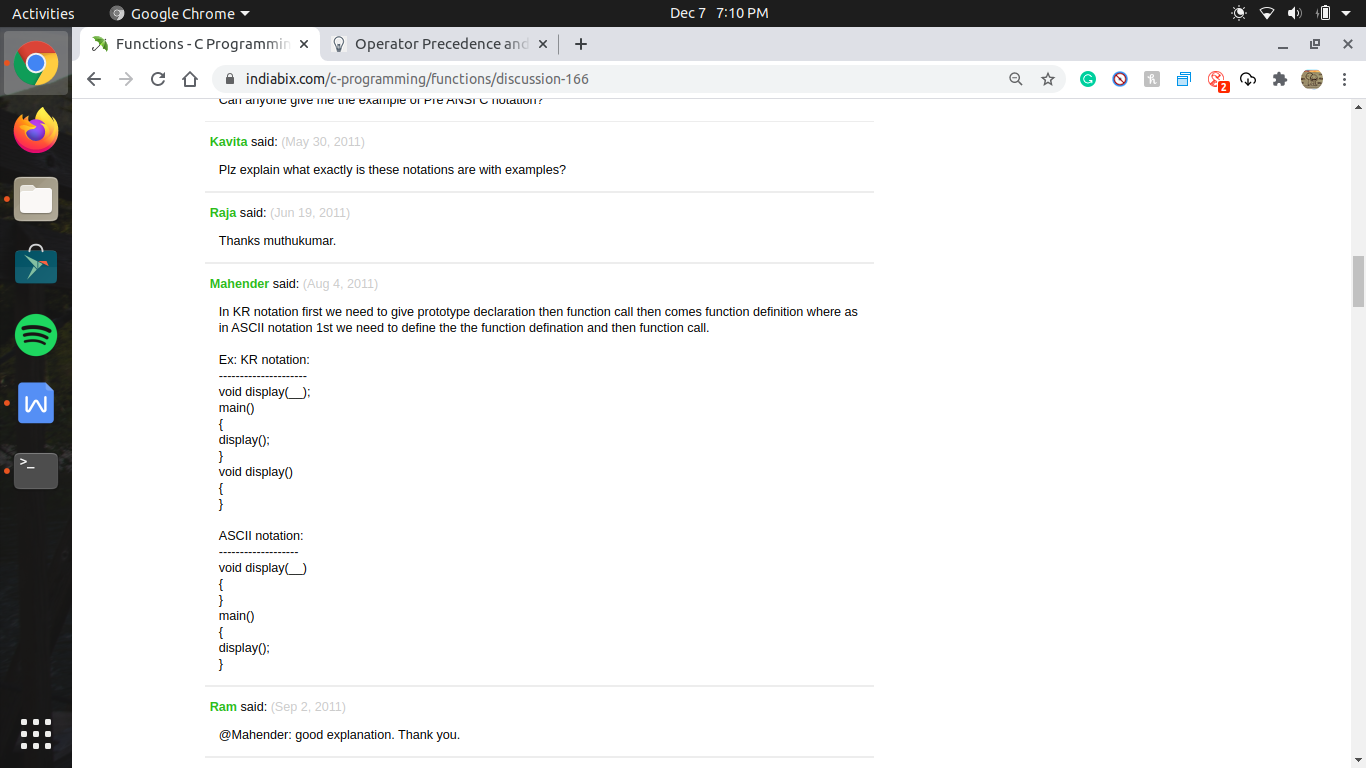
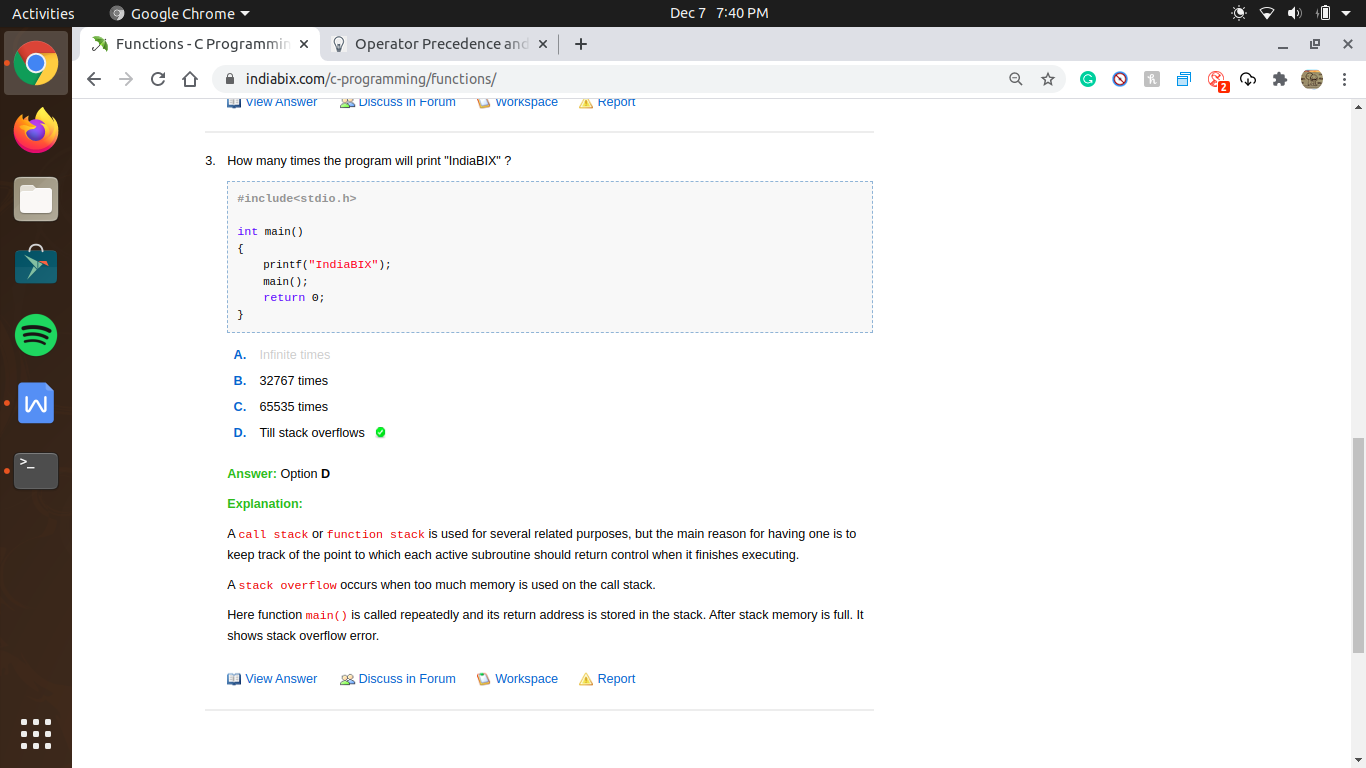
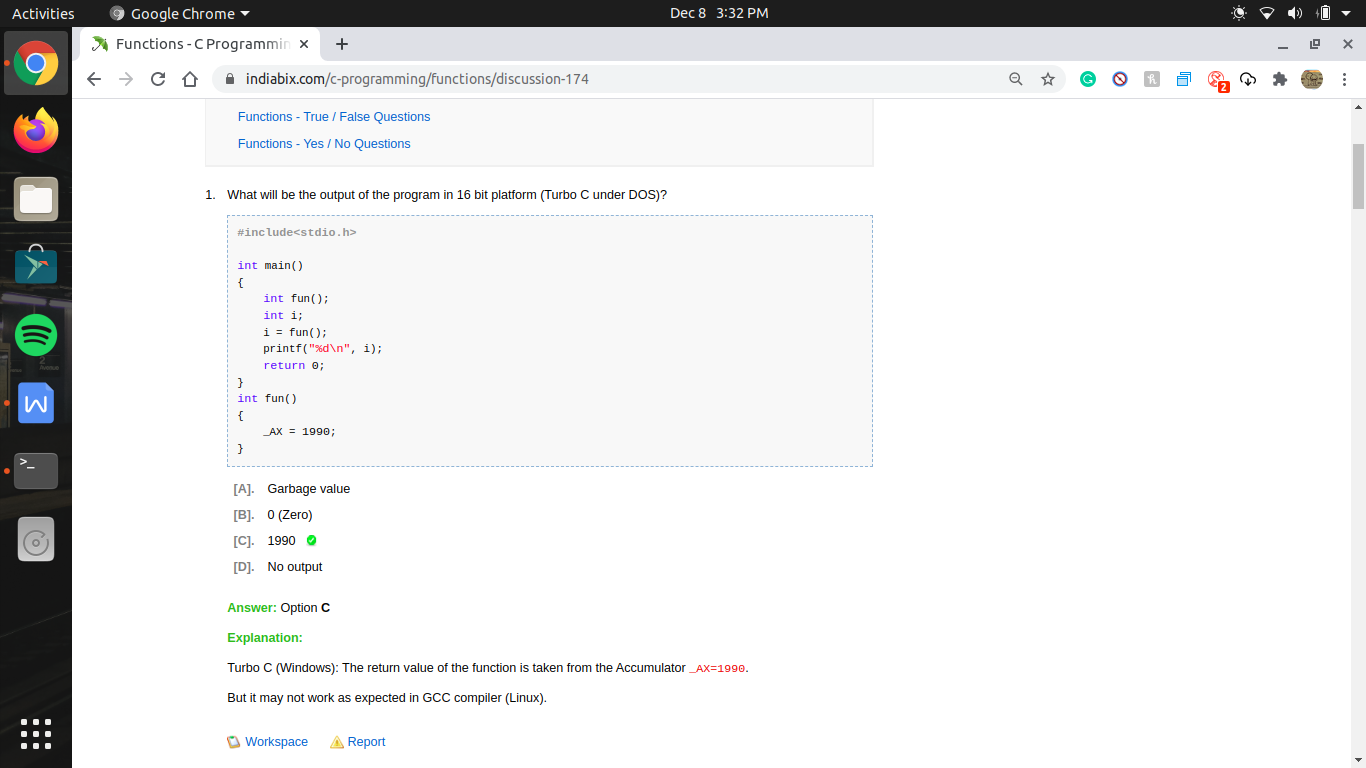
function mcq practice

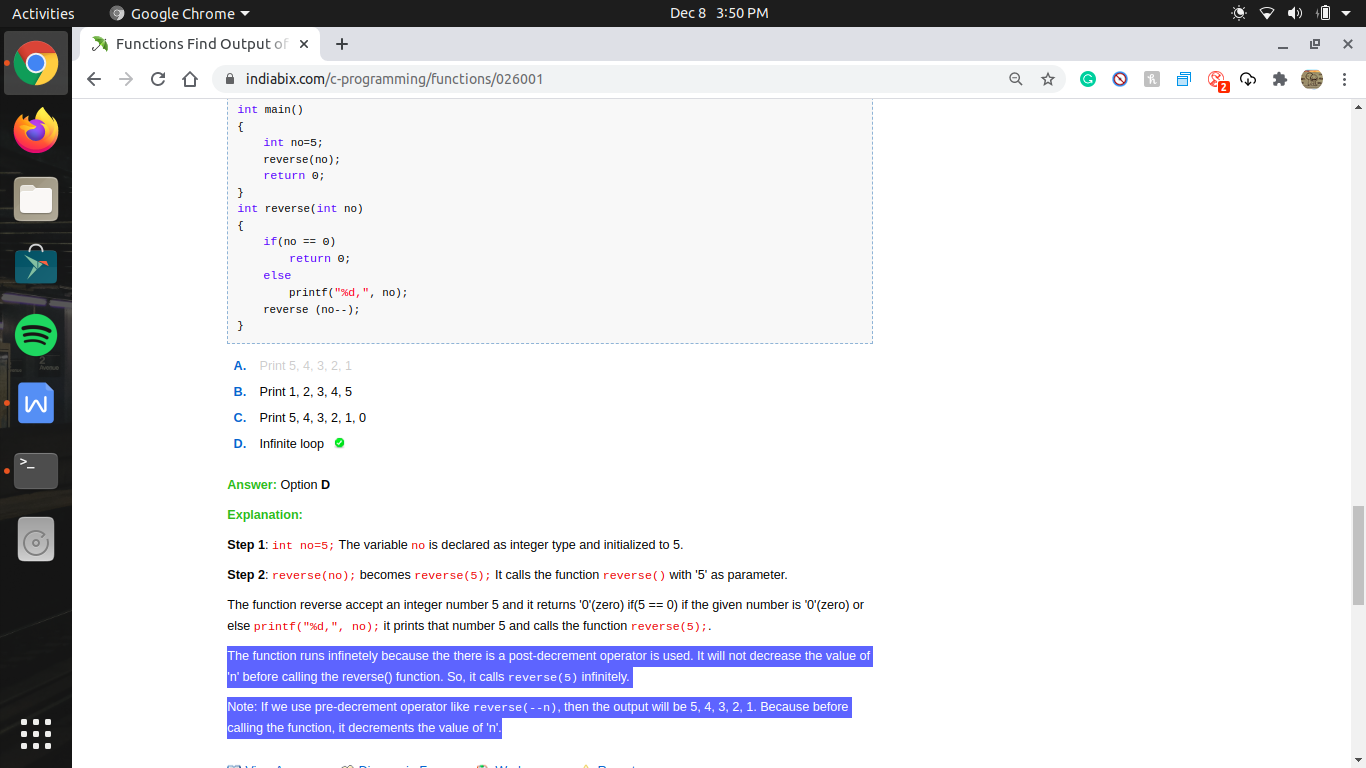


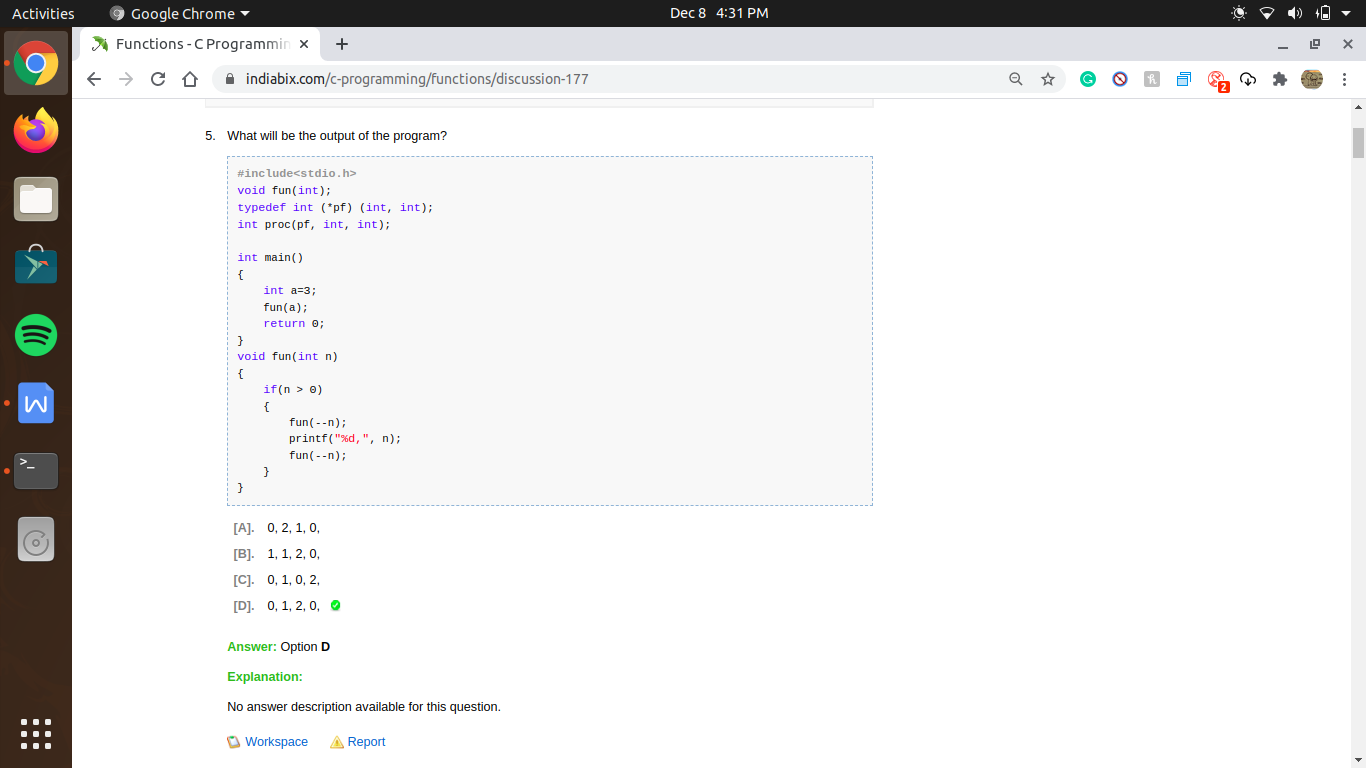




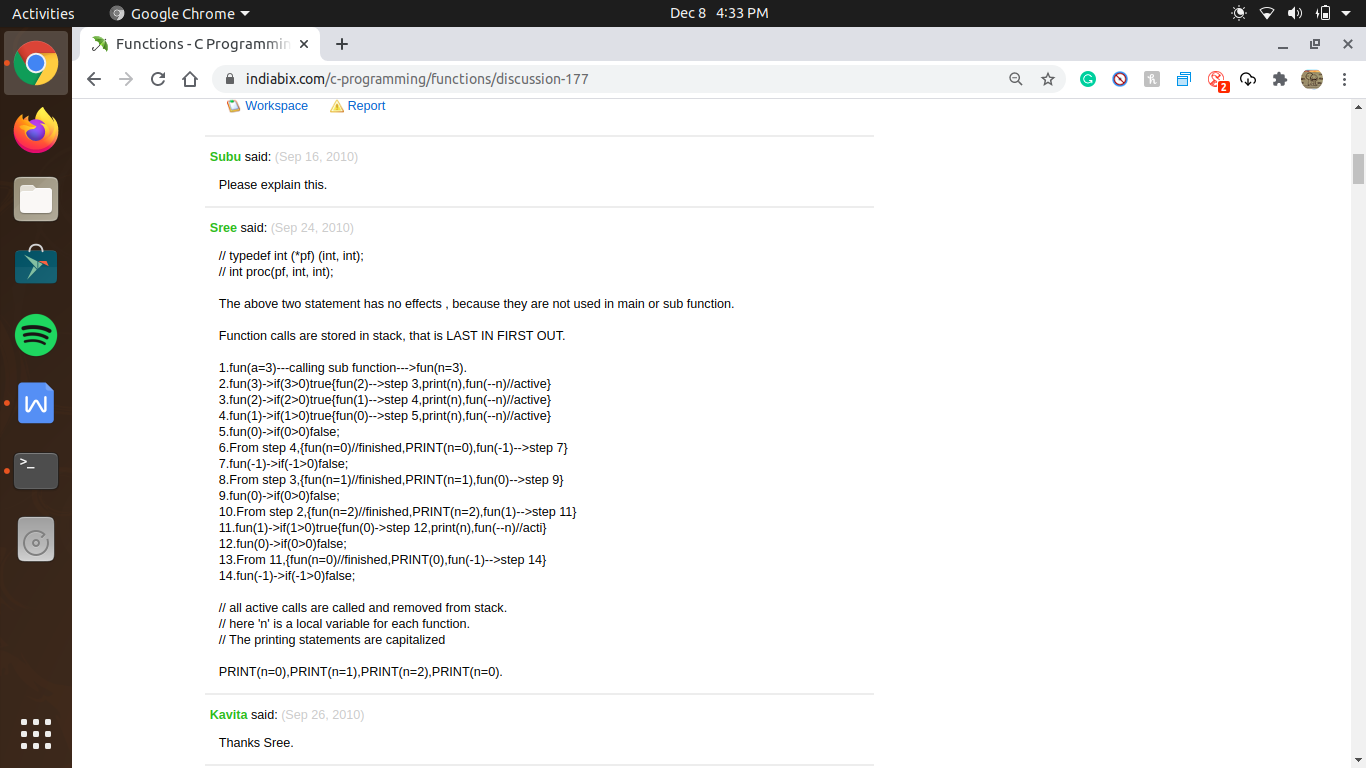


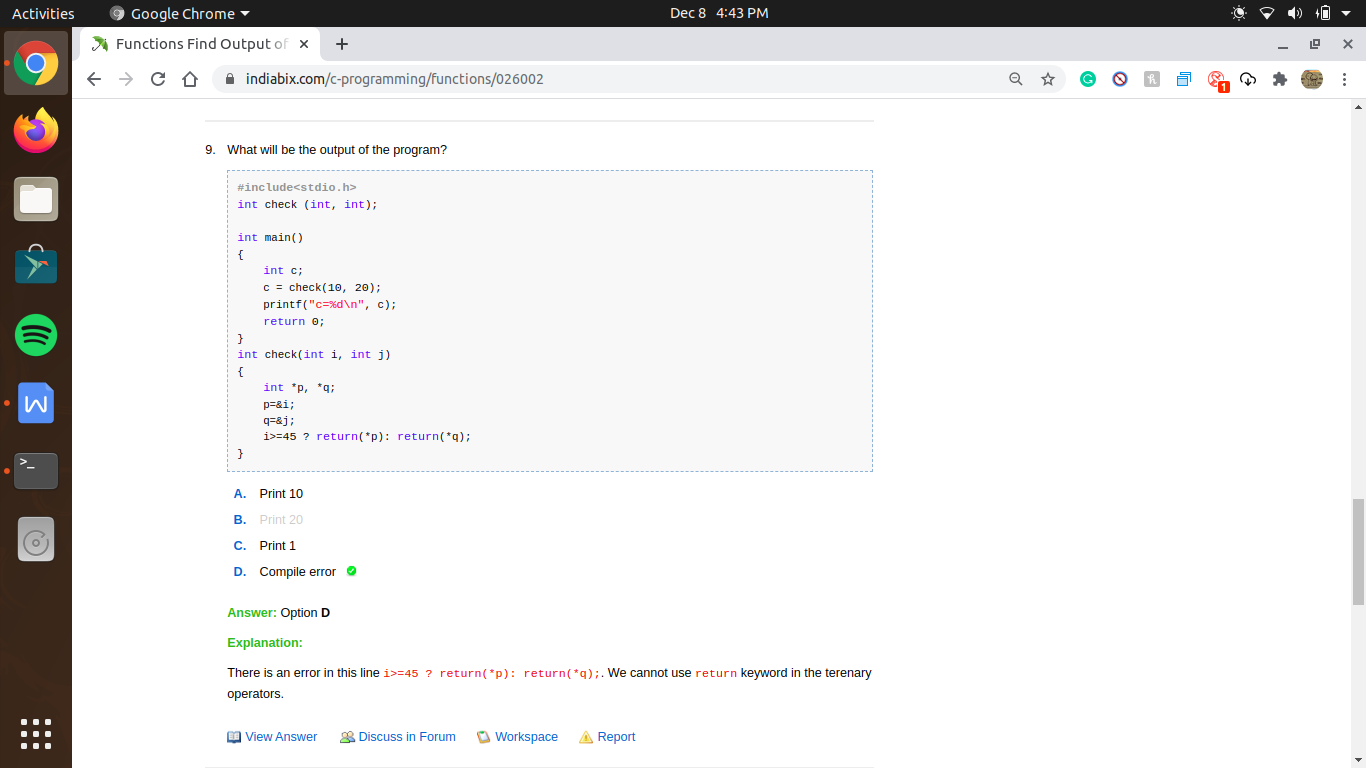
int fun(); as bracket indicates it is a function thus here we declare the function , as because we define the function after main();





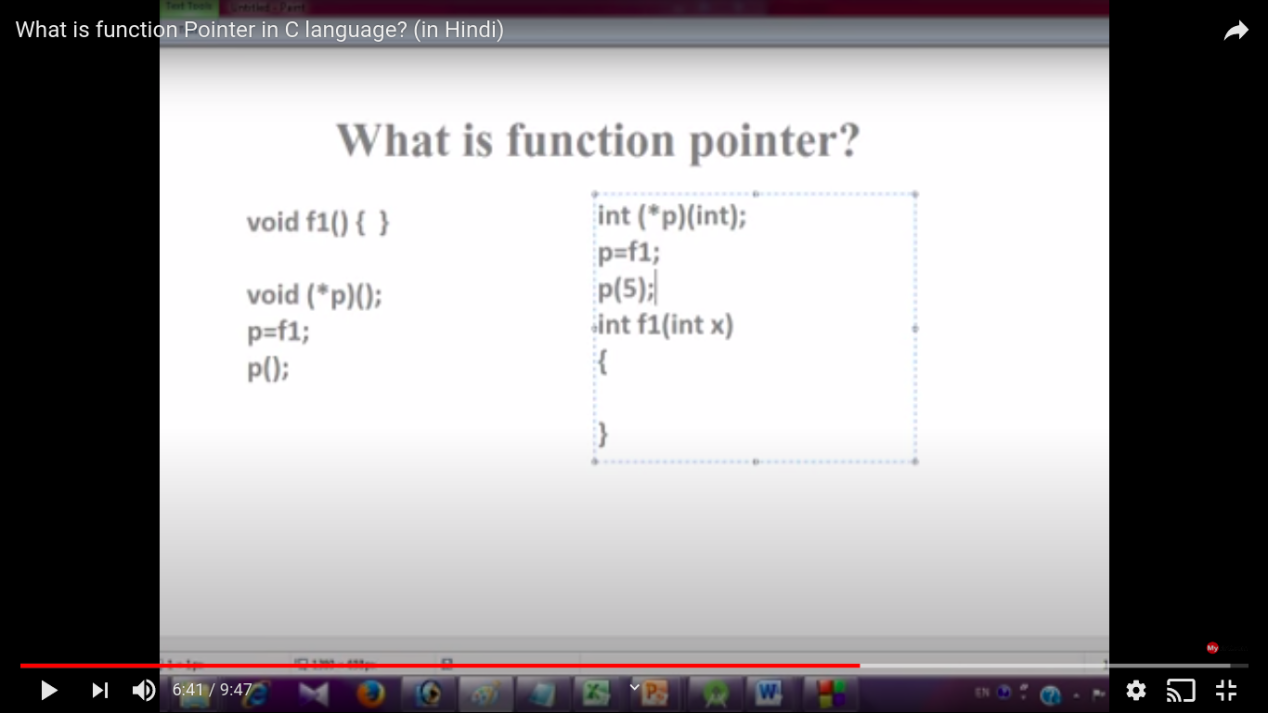
always solve recursion problem by making stack





we can’t use return function on terenary operator

function pointer ->



in function pointer

3

1

int (\* p)(int , int );

22

how to read and understand this statement ->

\* p waha show kr rha hai ki p is a pointer and kyu ki yeh parenthesis ke andar hai toh es ka mtlb yeh function hai

ab yeh age move karege , or ek or parenthesis me (int , int ) it means , yeh aisa function pointer hai jo ek aise function ko point karega jis ke 2 argument pass hoge int int type ke

ab hum peeche move karege or es se hame pata chalega ki ki yeh ek function pointer hai jo aise function ko point karega jis me 2 argument hoge int type ka or sath me wo function int type ki value return karega

now we see difference on

2

int \* p( int , int);

1

as here no bracket is there so first we move forward

and say p in a type function (not function pointer )

which passing argument as (int , int)

then we move back and say which return address of data type int

i.e we treat is as (int \*) p(int , int);

one more thing on function pointer

agar hame function pointer ke andar kisi function ka address store karana hai toh us time hum funtion ke bad parenthesis nhi lagayege

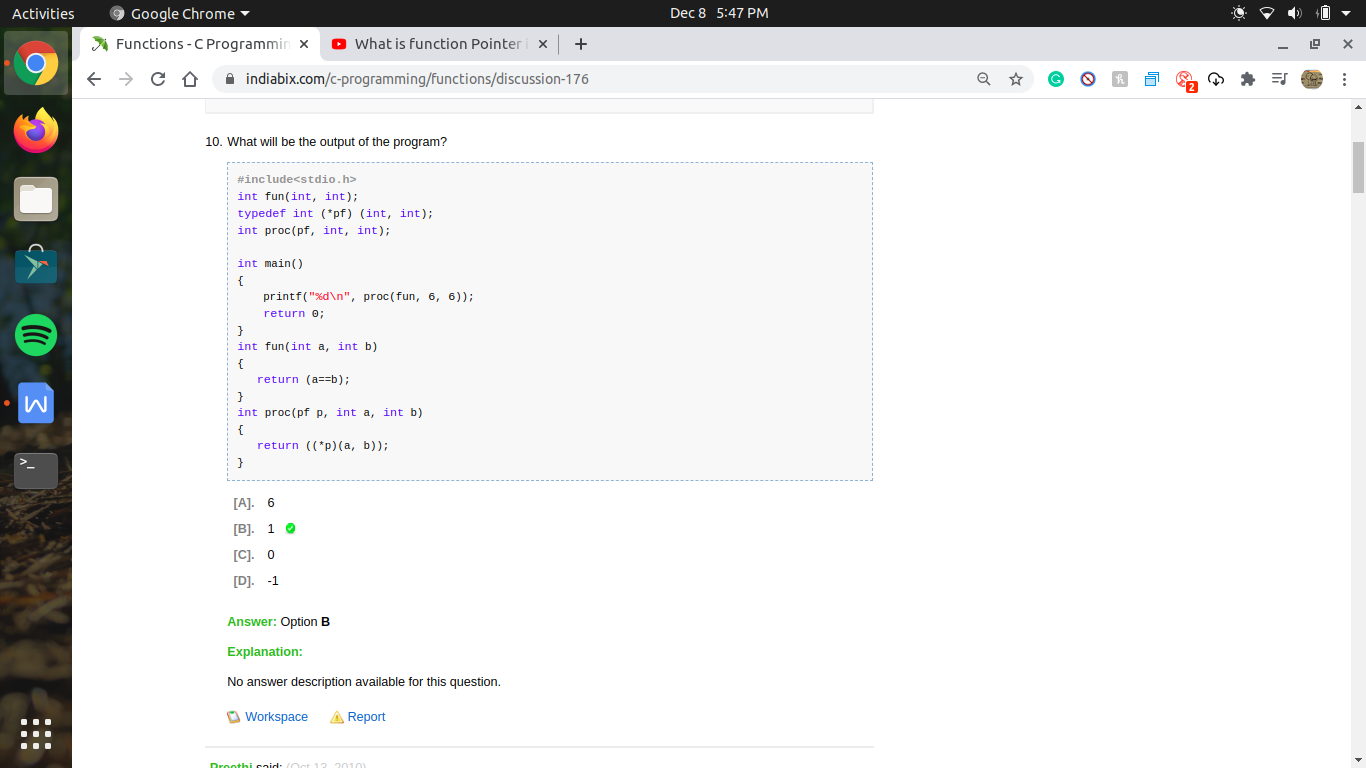
i.e

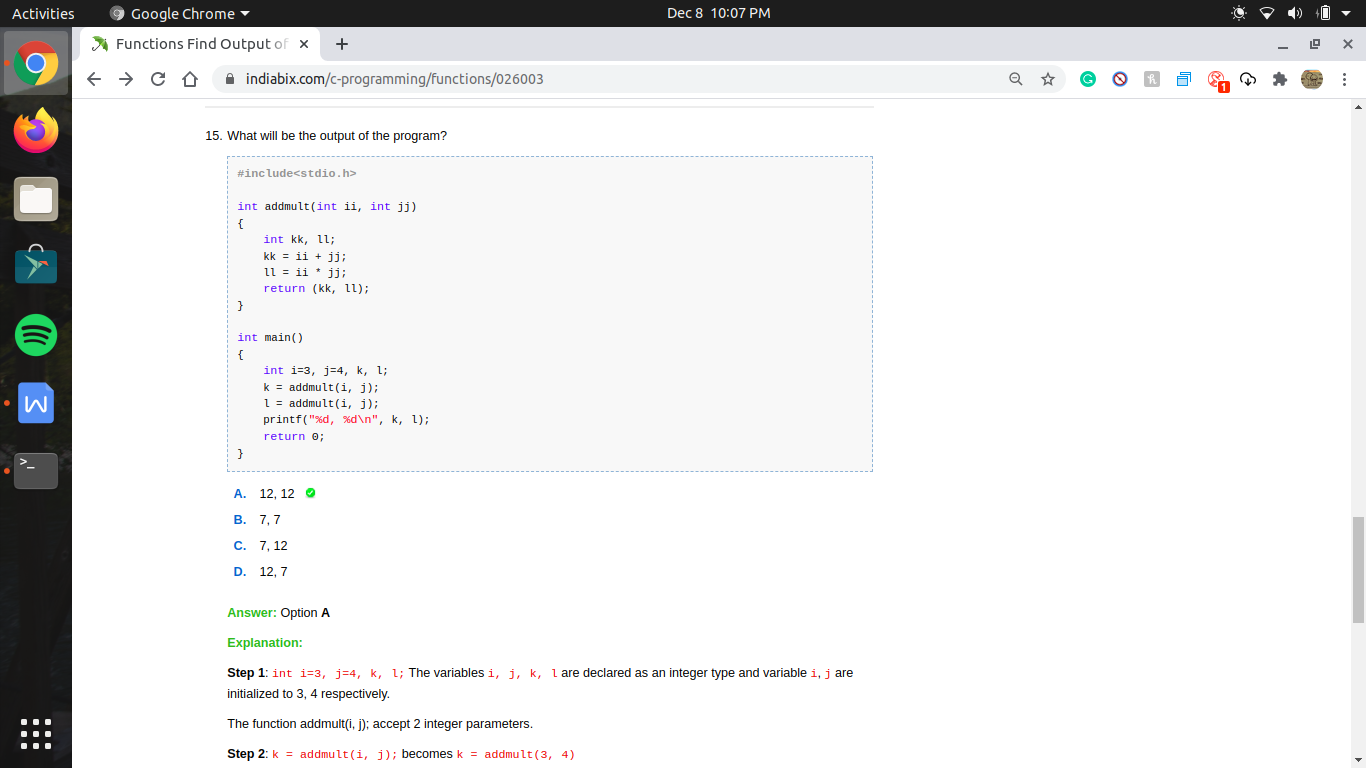
p=f1; //no need to apply parenthesis

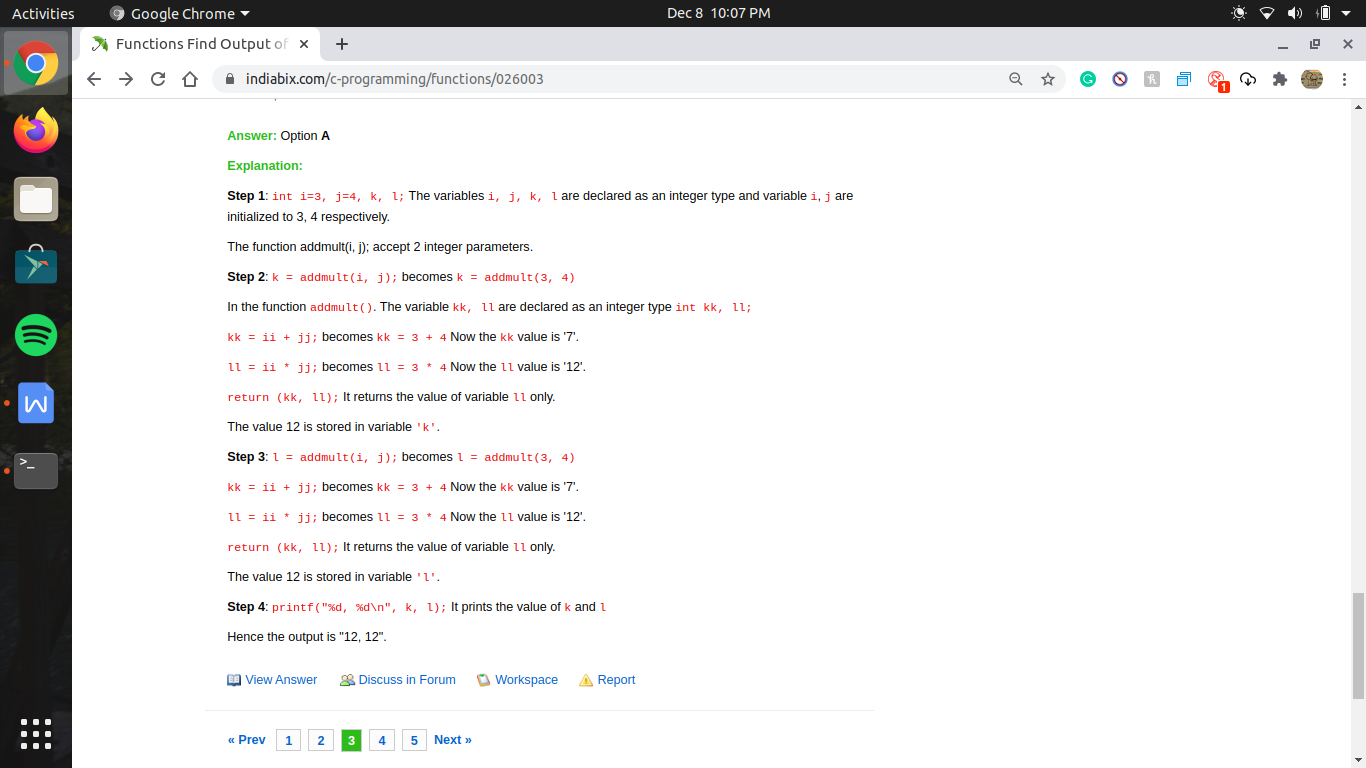
ab call karane ke leye

hum pointer ko aise treat karege jaise us function ka naya nam rkh diya gaya hai

i.e p(int ,int); //here f1(int ,int ) funct treat same as p(int ,int);



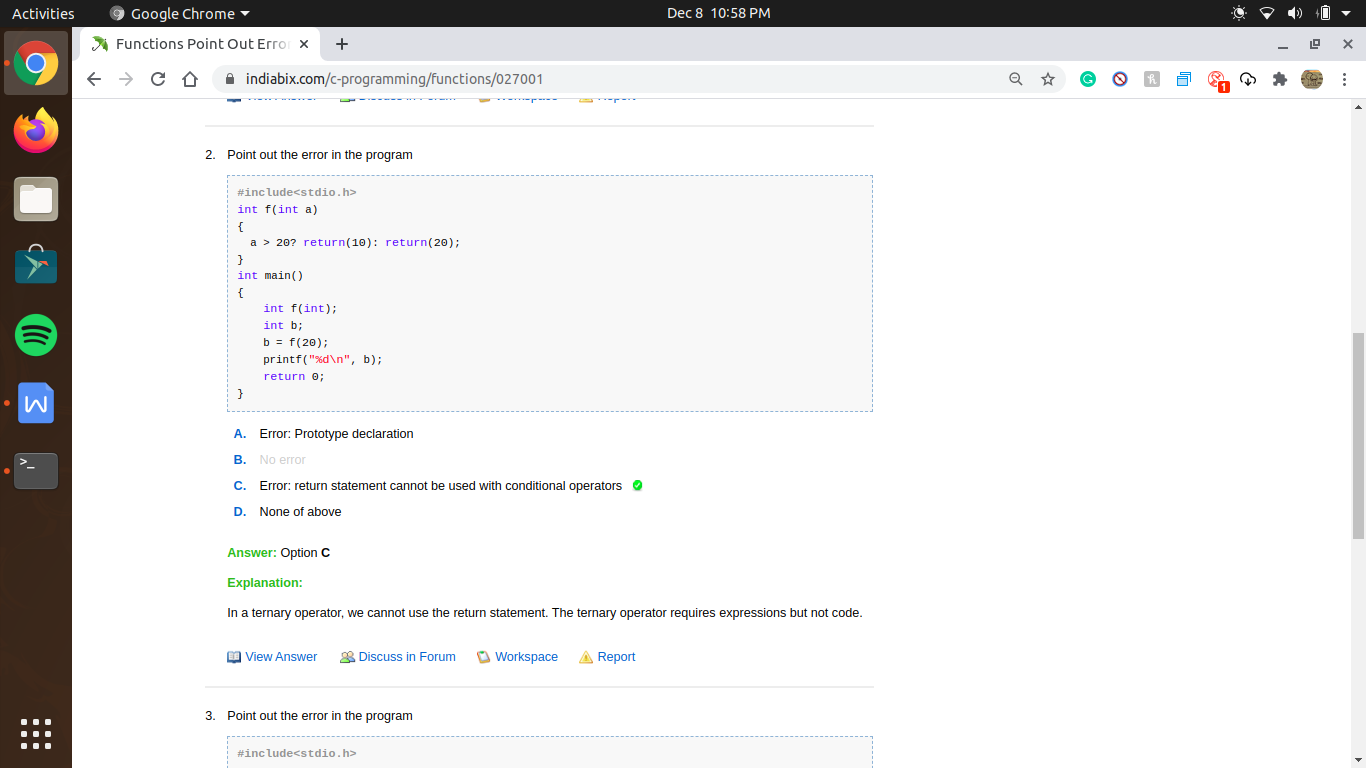




Comma (,) operator has left to right associativity, so while returning the rightmost operator will be returned and no error because of this.

i.e return (a,b); thus means

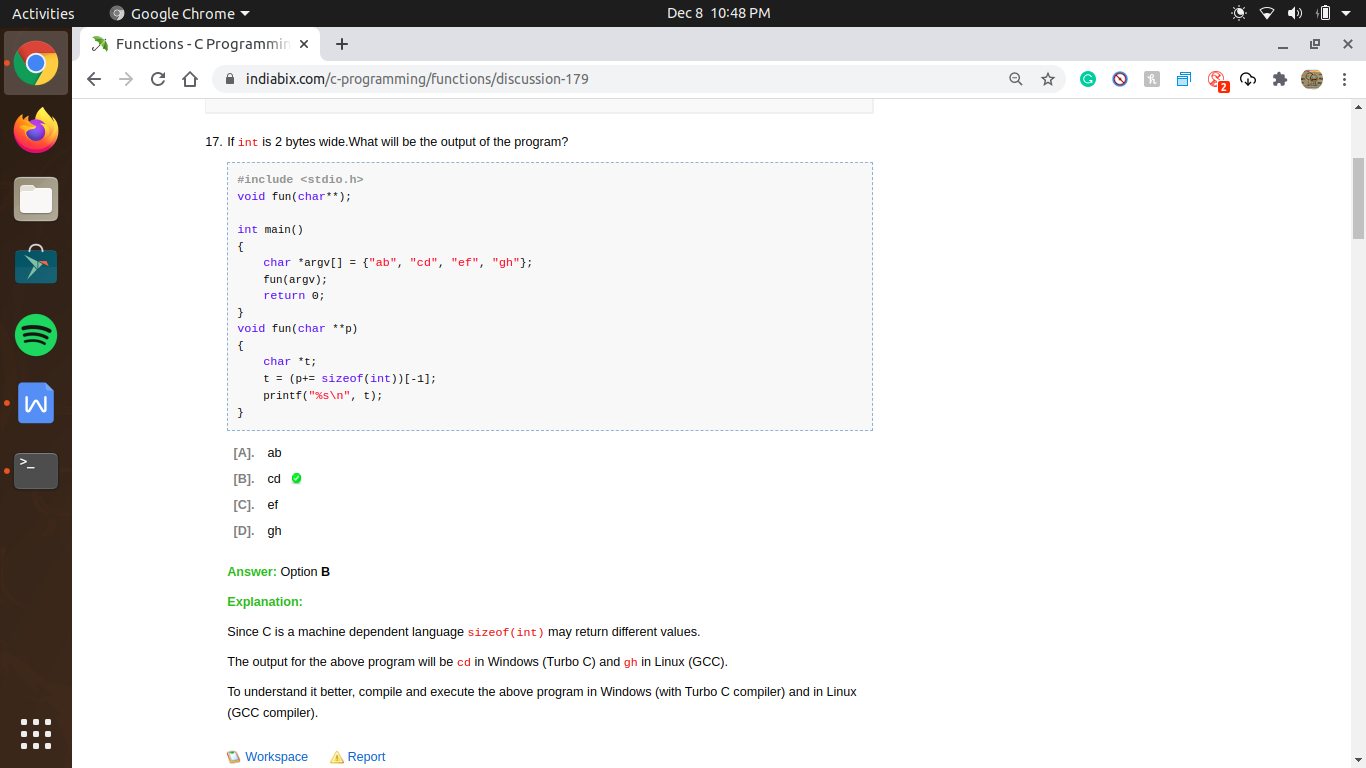
function will return only b and ignore a (i.e) most right hand side

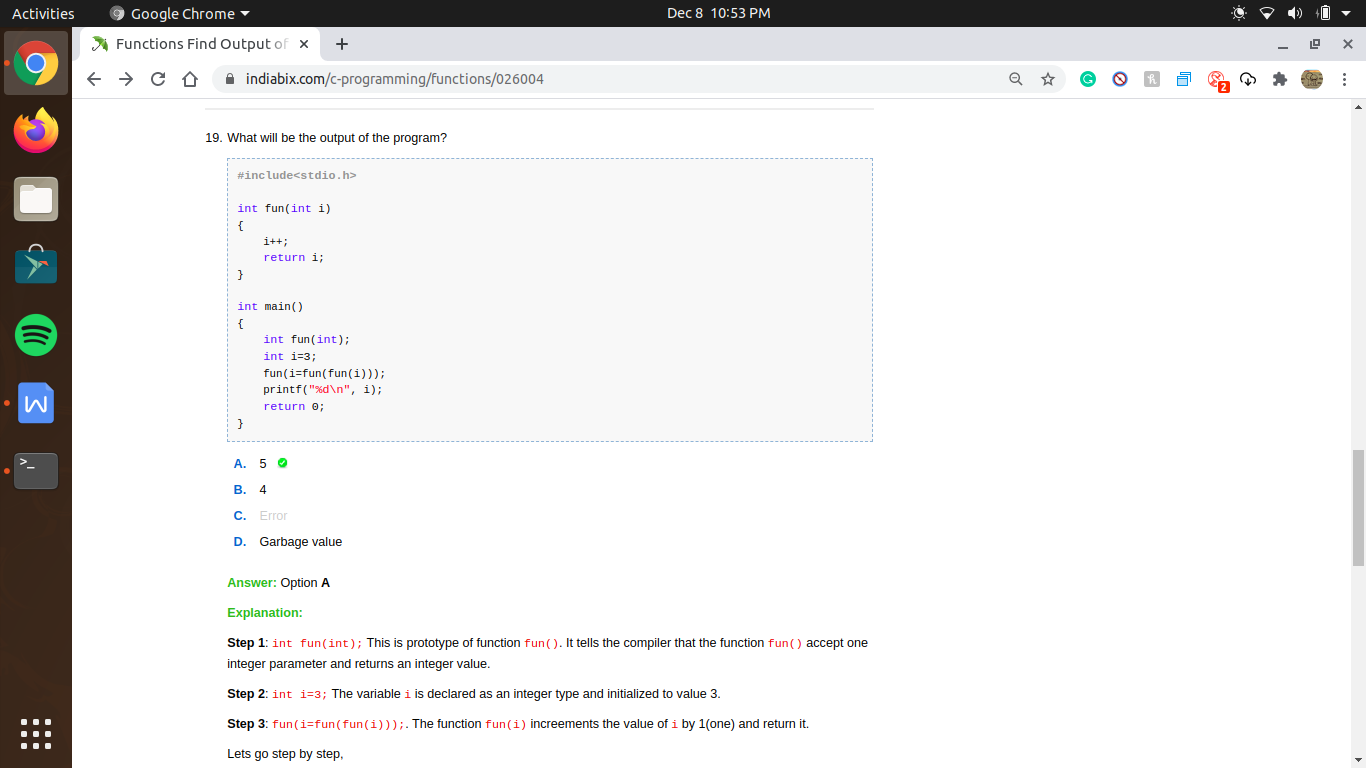


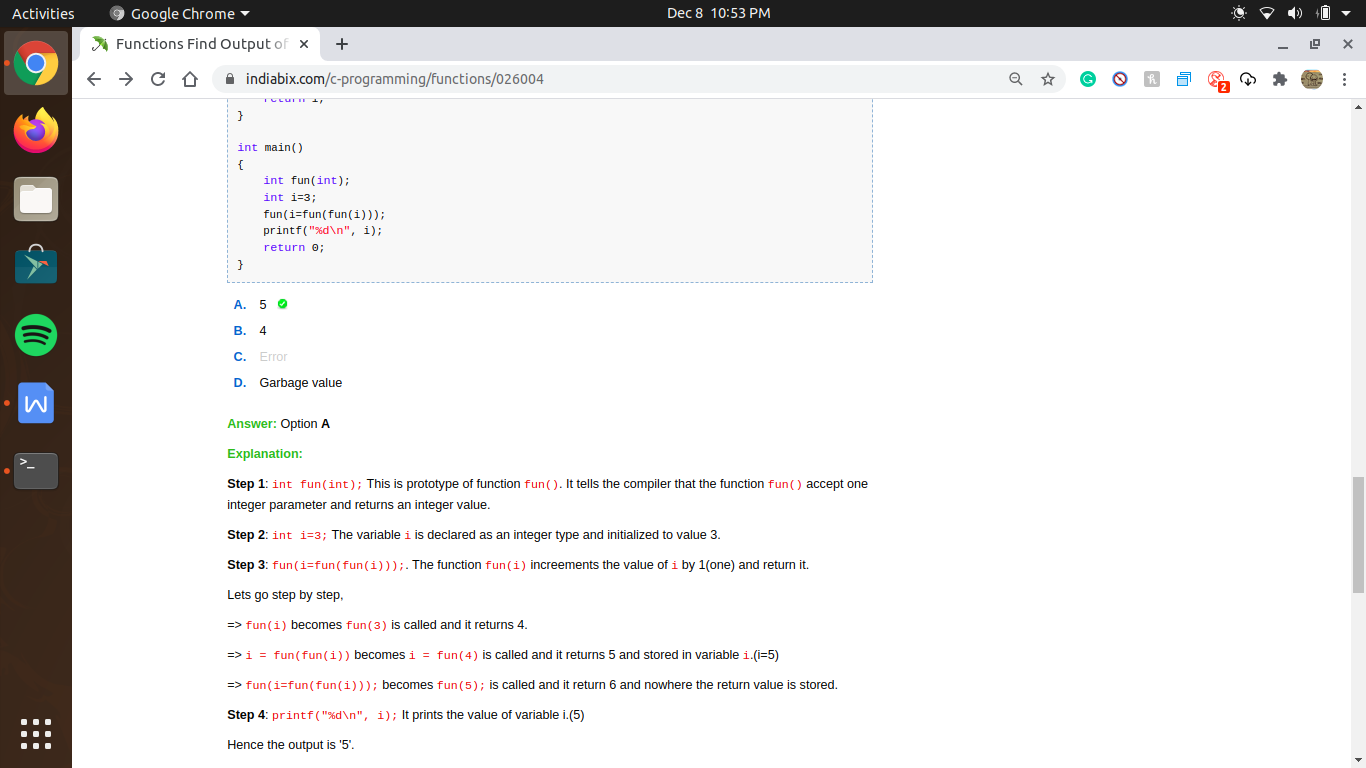
overload operator ->

x+=10 , this means x=x+10;

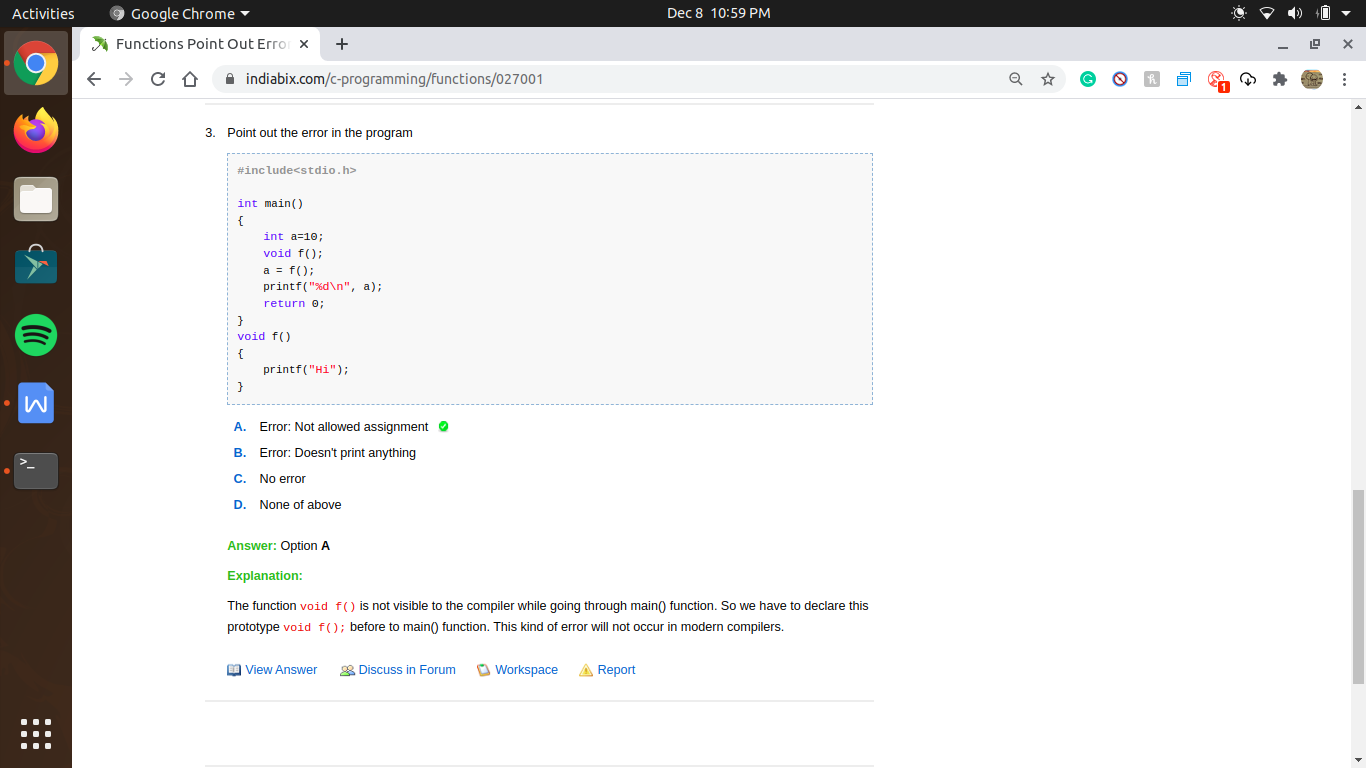
x-=10 , this means x=x-10;

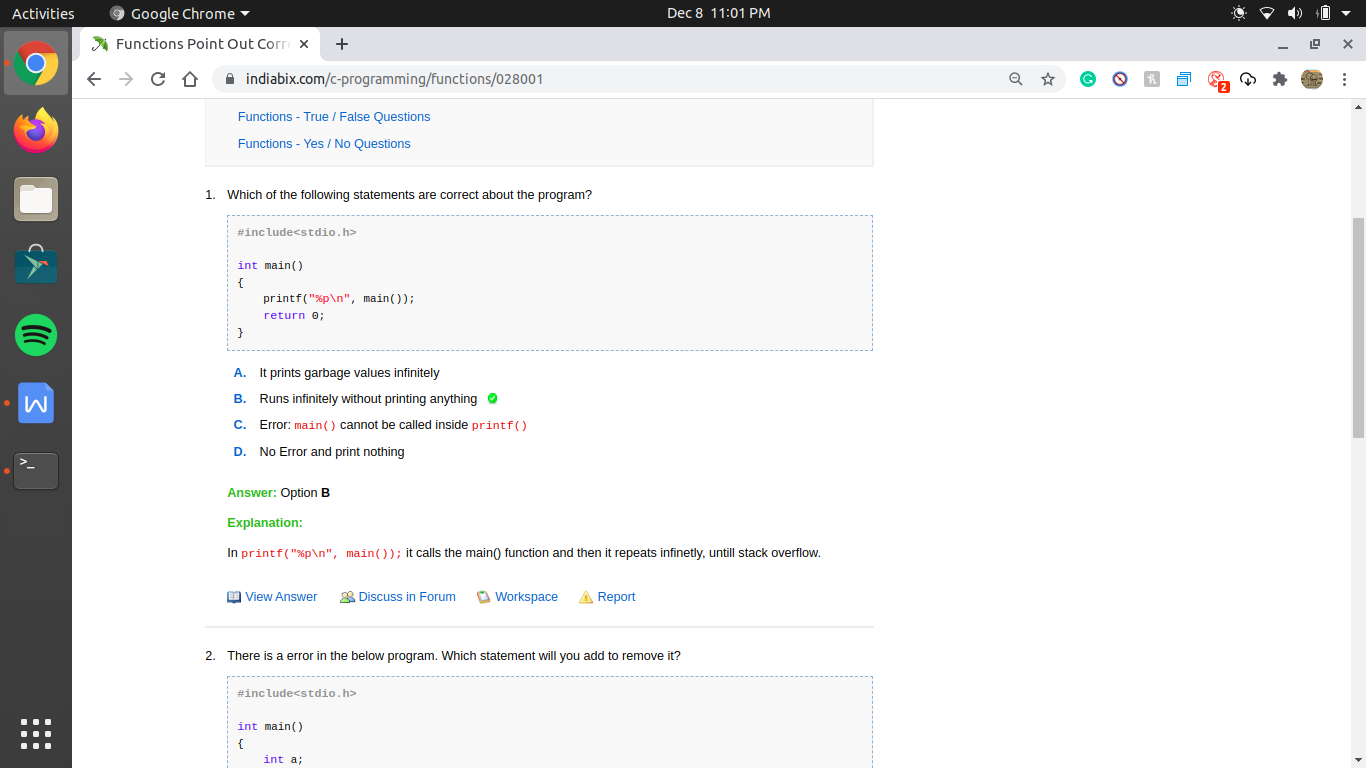


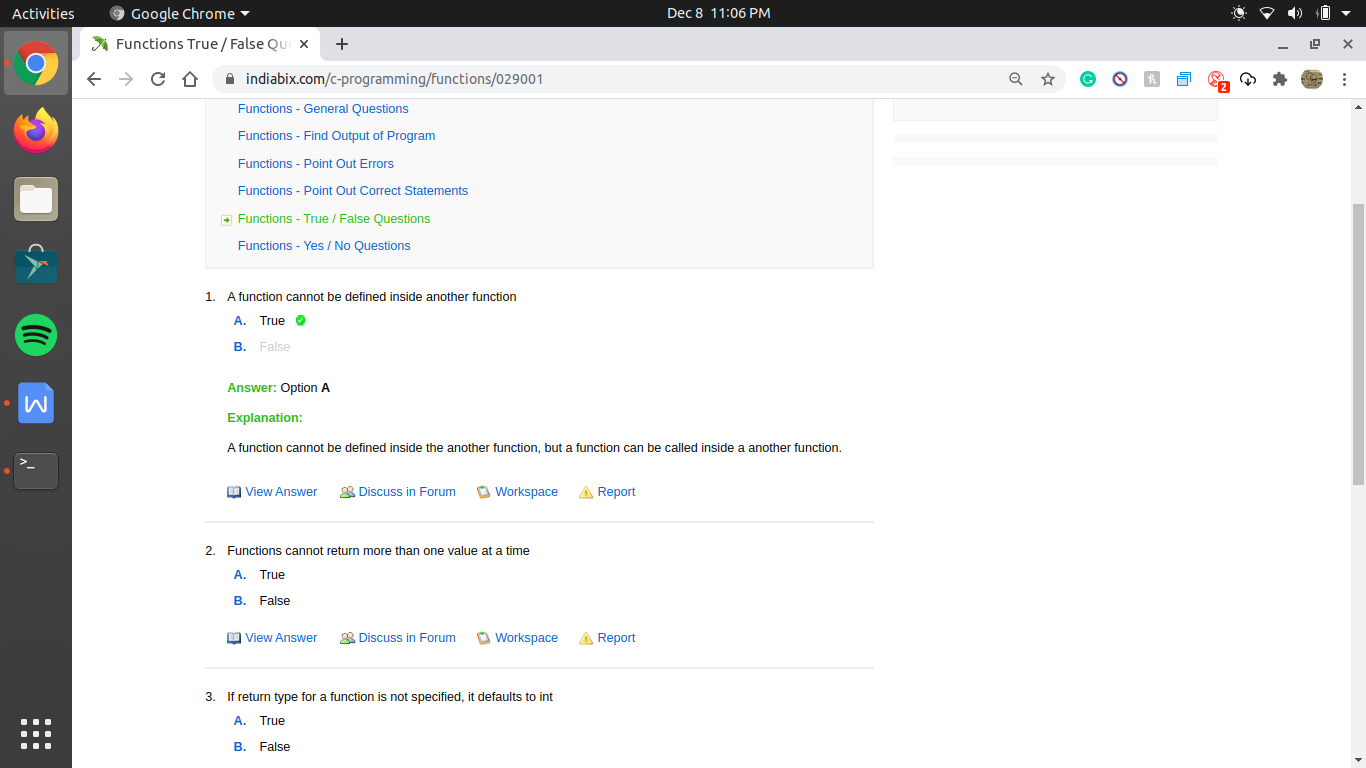




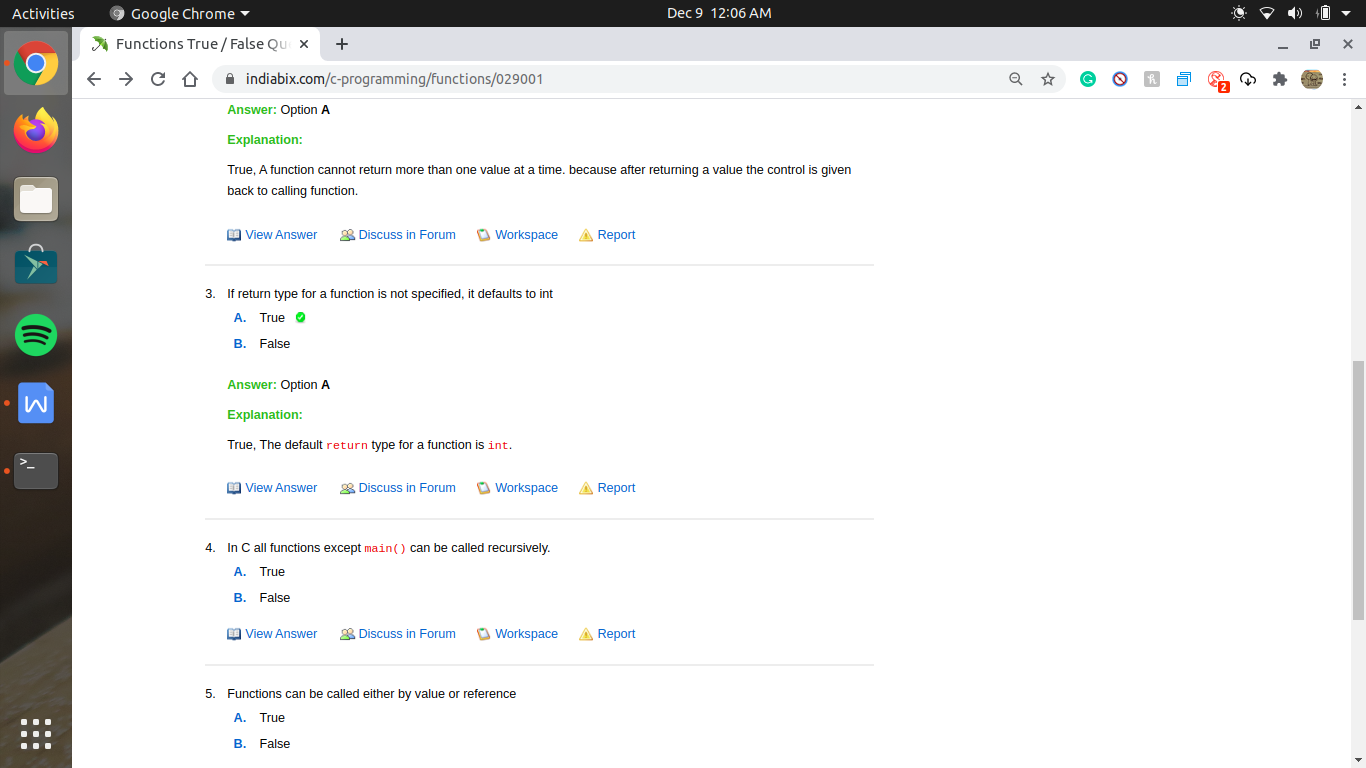
variable inside a function can’t used by another function , thus i variable in main or i variable in other function , they both only name is common , both are different variable

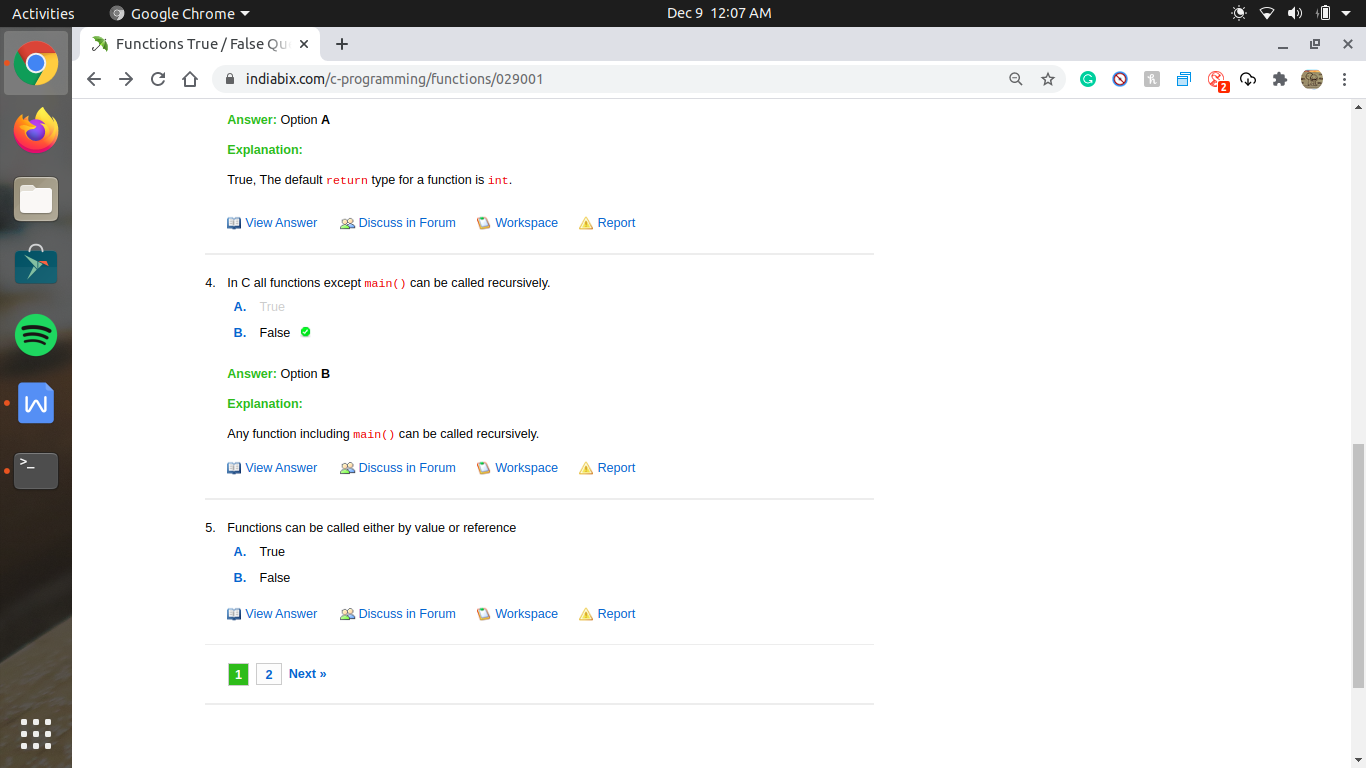


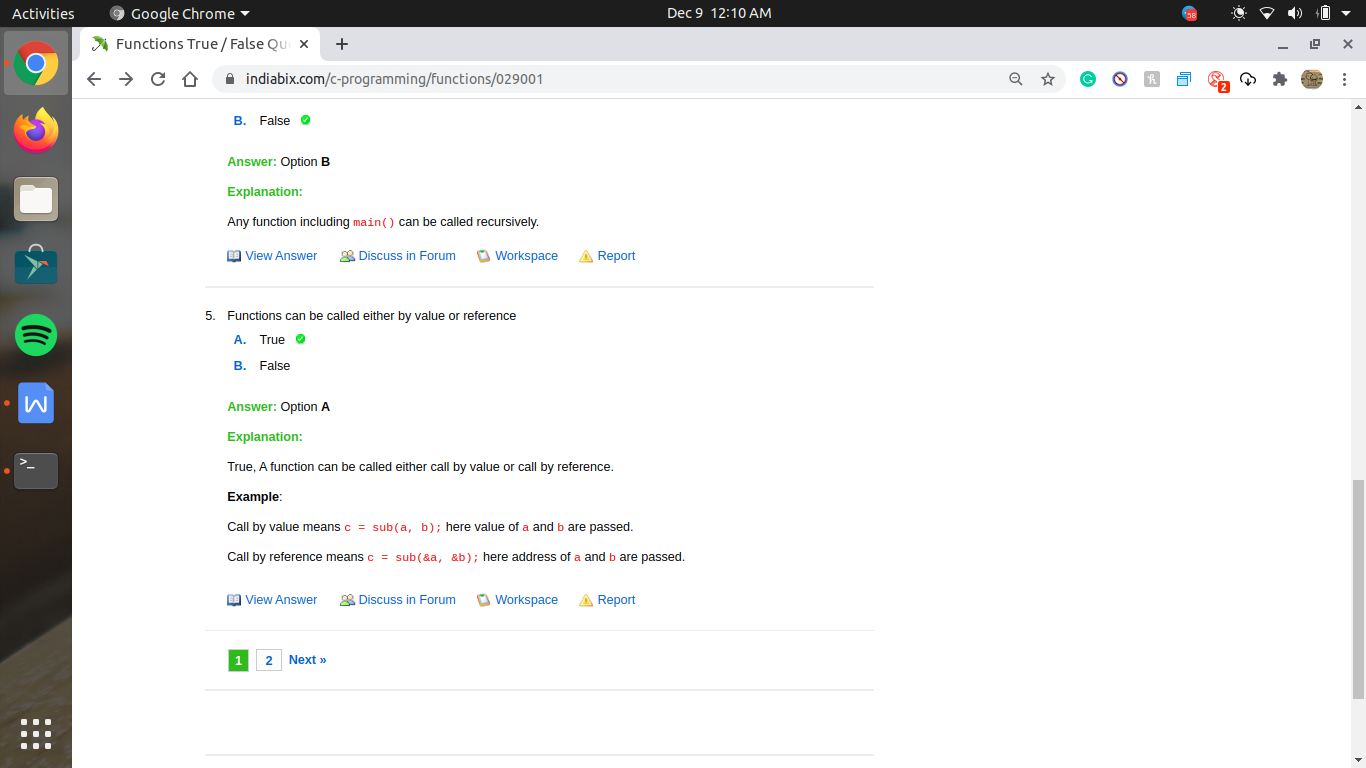


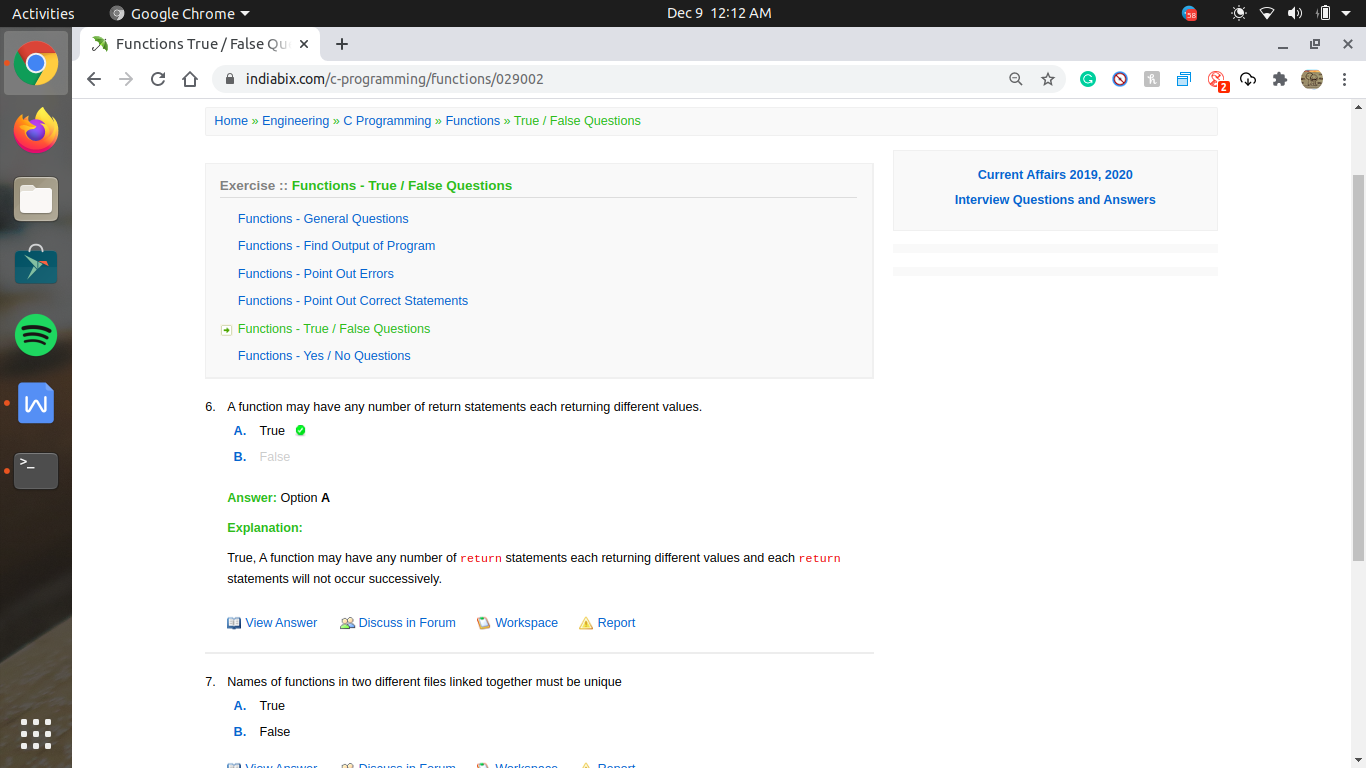


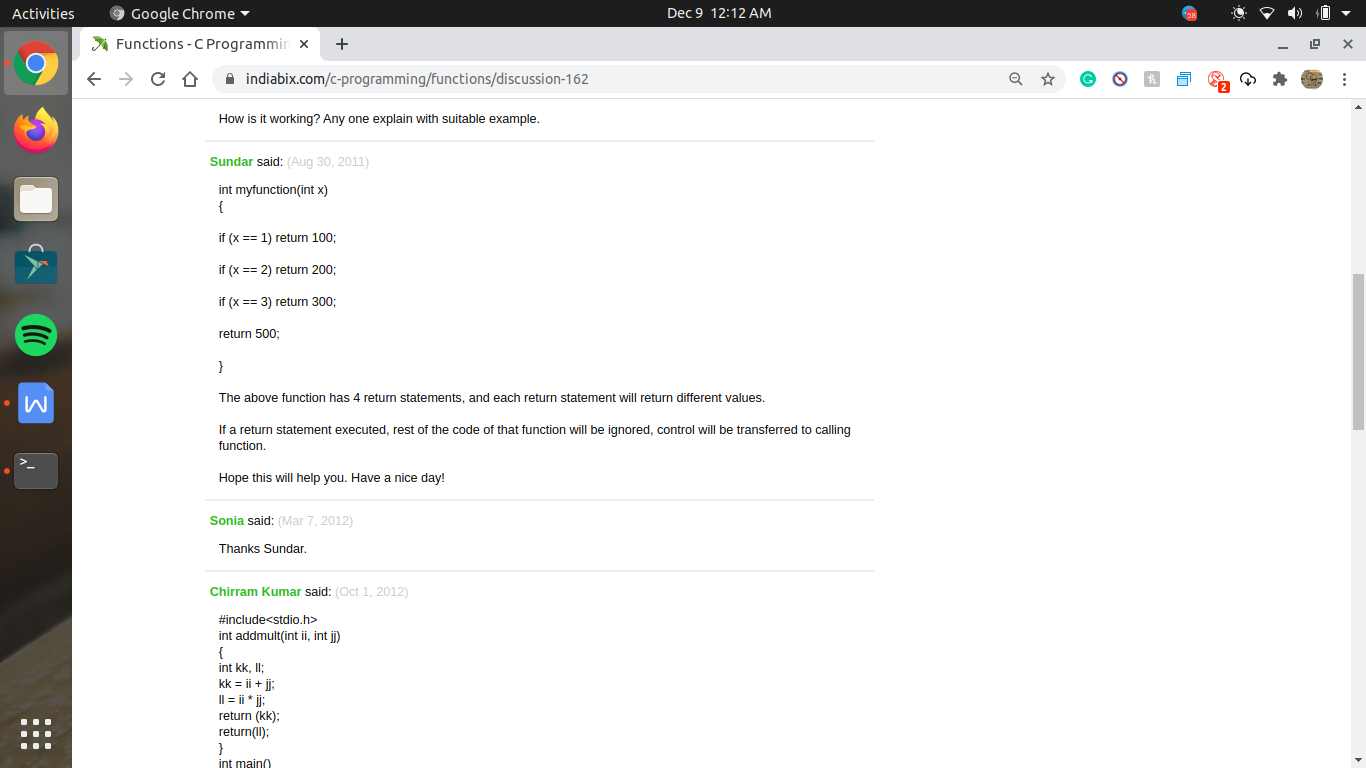
In typical C language a function cannot be defined inside an another function.  
  
But, in some modern compilers (GCC in Linux) it is possible, lets see an example.









No, In a function two or more return statements can occur but not successively.

