PYTHON PROGRAMMING
GATE DA/DSA

Agenda: Gate PYQs Practice.

GATE 2024

```
def g(p: int) -> int:
    print(p)
    return p

def h(q: int) -> int:
    print(q)
    return q

def f(x: int, y: int) -> None:
    g(x)
    h(y)
```

p is type int → input argument. int: return type.

f(g(10), h(20)) ⟶ f(10, 20) (x = 10, y = 20)

10 ✓
20 ✓

Soln: f(g(10), h(20))

g(10)
h(20)

g(10) → 10 ✓ (1)

h(20) → 20 ✓ (2)

f(10, 20)

g(10) → 10 ✓ (3)
h(20) → 20 ✓ (4)

O/P: 10 20 10 20

GATE 2024:

```
def fX():
    a = input()
    if "n" not in a:
        fX()
    print(a)
fX()
```

Q. Assume that the input to the program from the command line is

given in following order each time.

→ 1

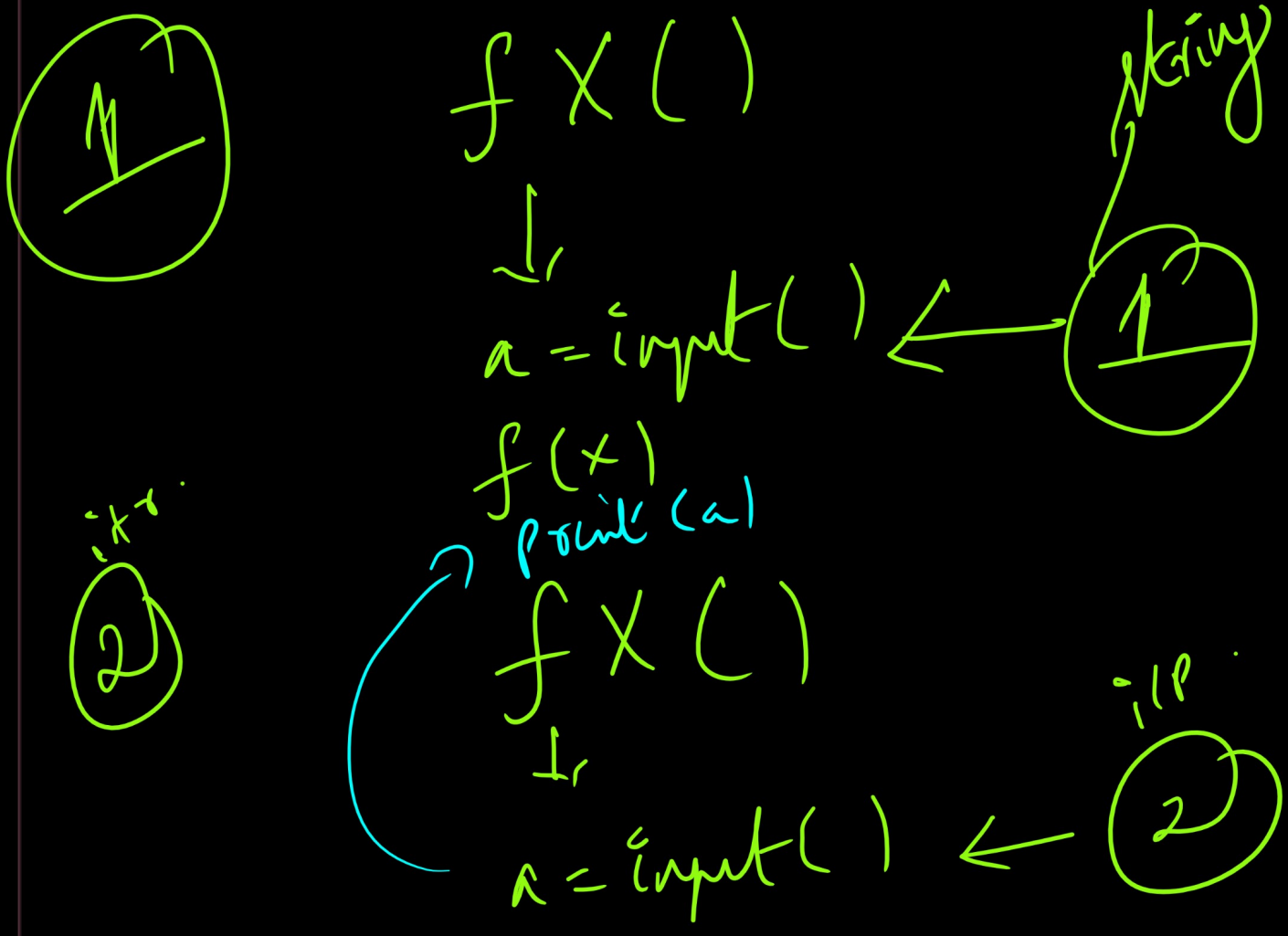
→ 2

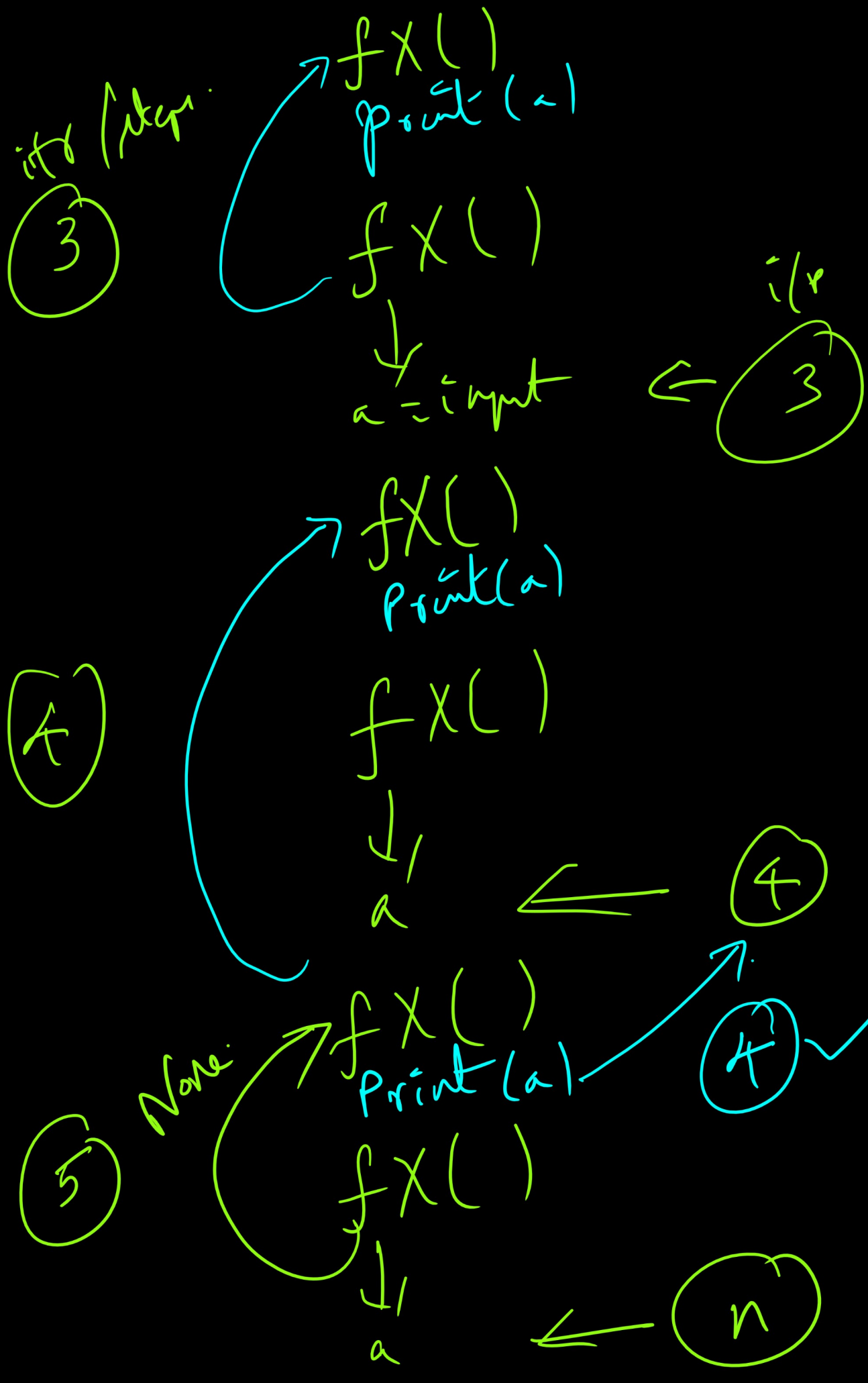
→ 3

→ 4

→ n

What will be the output?




int (int)
3
fX( )
print(a)
fX( )
a = input
input
3
fX( )
print(a)
4
fX( )
a
4
4
None
5
fX( )
print(a)
fX( )
a
n

4 3 2 1

a) 1 2 3 4 n

b) 1 2 3 4

c) 4 3 2 1 n

d) 4 3 2 1

GRACE 2023

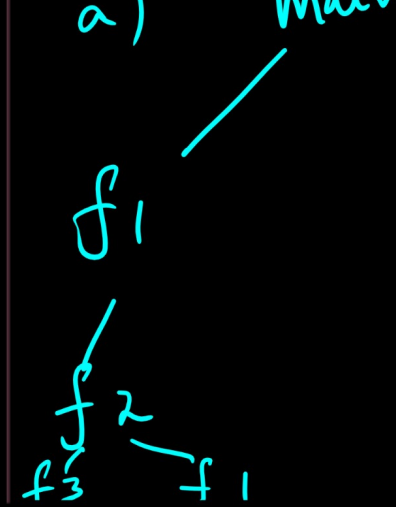
```
def f1() -> int:
    return 1
def f3() -> int:
    return 5
```

```
def f2(X: int) -> int:
    f3()
    if X == 1:
        return f1()
    else:
        return X * f2(X-1)
```

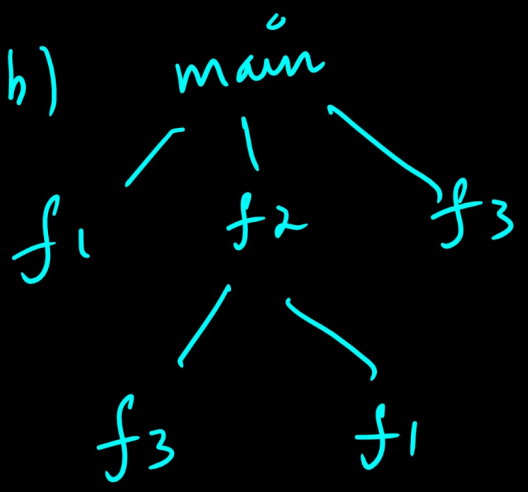
```
f1()
f2(2)
f3()
```

Which one of the following options represents the activation tree corresponding to the main code?

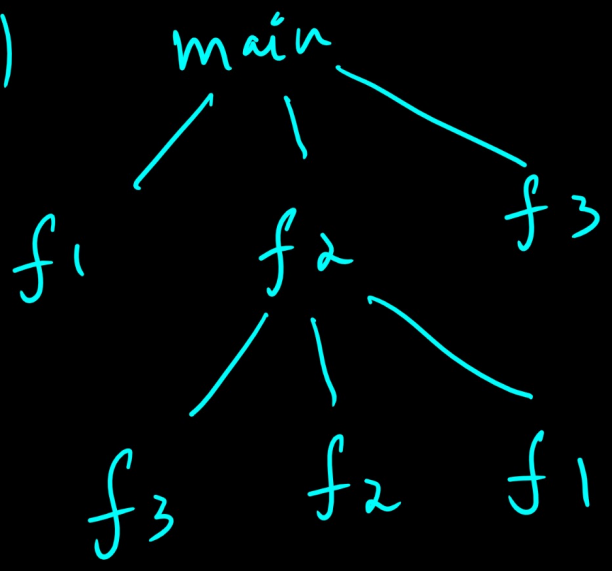
a)



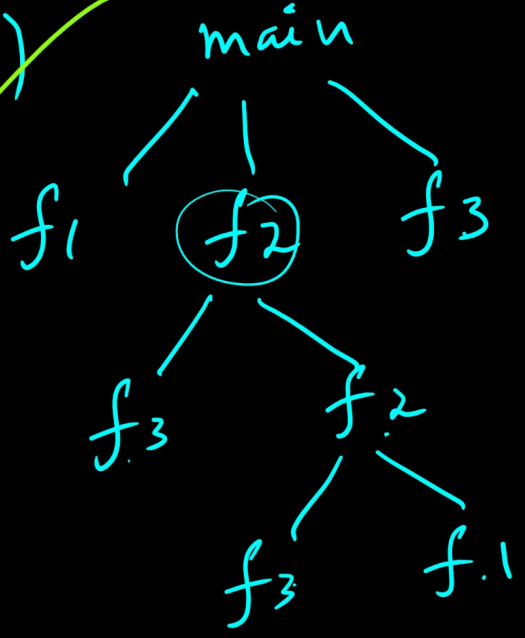
b)



c)



d)



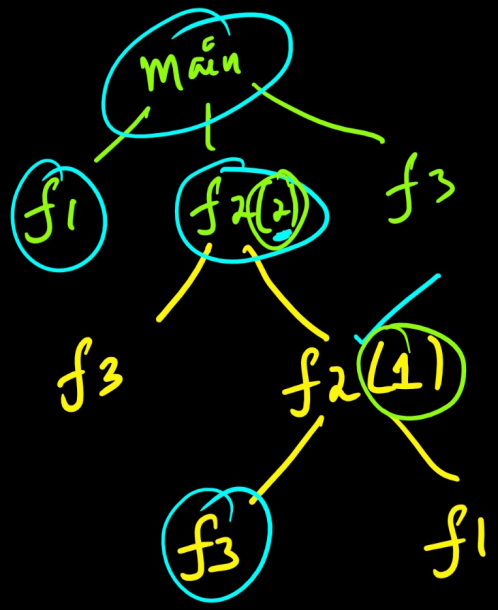

```
def f1():
  return 1

def f3():
  return 5

def f2(X):
  f3()
  if X==1:
    return f1()
  else:
    return X*f2(X-1)

f1()
f2(2)
f3()
```

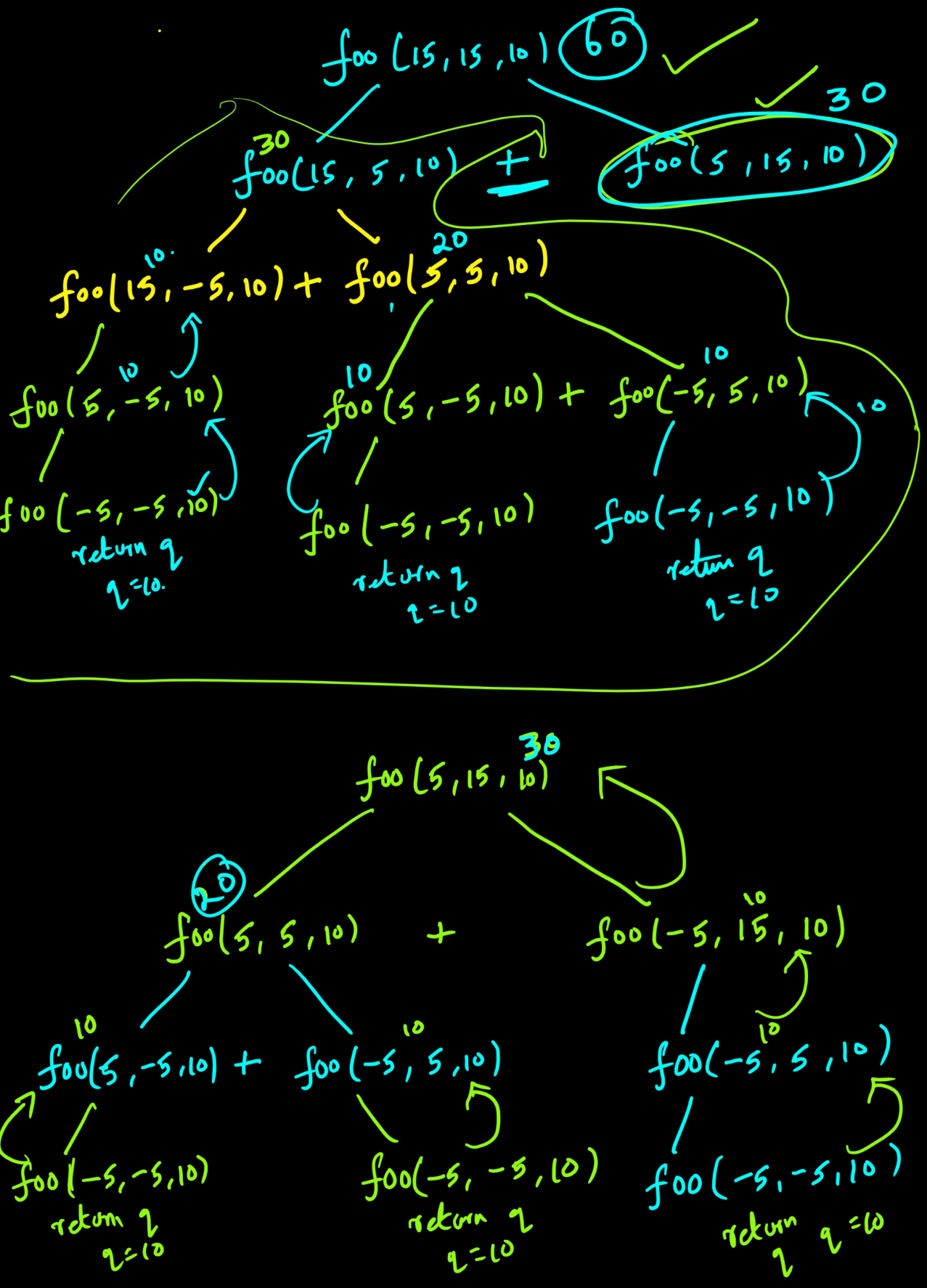
f2(2) → return X*f2(X-1) → f2(2-1)



```
def foo(x: int, y: int, q: int)->int:
  if ((x<=0) and (y<=0)):
    return q
  if x<=0:
    return foo(x, y-q, q)
  if y<=0:
    return foo(x-q, y, q)
  return foo(x, y-q, q)+foo(x-q, y, q)

r = foo(15, 15, 10)
print(r)
```

a) 20
b) 220
c) 160
d) 100
e) 60
f) None of these
g) Error.


foo(15, 15, 10) 60
30
foo(15, 5, 10) +
foo(5, 15, 10)
10
foo(15, -5, 10) + foo(5, 5, 10)
20
10
foo(5, -5, 10)
10
foo(5, -5, 10) + foo(-5, 5, 10)
10
foo(-5, -5, 10)
return q
q=10
foo(-5, -5, 10)
return q
q=10
foo(-5, -5, 10)
return q
q=10
30
foo(5, 15, 10)
20
foo(5, 5, 10) +
foo(-5, 15, 10)
10
10
foo(5, -5, 10) +
foo(-5, 5, 10)
10
foo(-5, 5, 10)
foo(-5, -5, 10)
return q
q=10
foo(-5, -5, 10)
return q
q=10
foo(-5, -5, 10)
return q
q=10

GATE 2021

```
def simpleFunction(Y: list, n: int, x: int)->int:
  total = Y[0]
  for loopIndex in range(1,n):
    total = x*total+Y[loopIndex]
  return total
```

Let Z be an array/list of 10 elements with Z[i] = 1 for all i such that $0 \le i \le 9$. The value returned by simpleFunction(Z,10,2) is ____

simpleFunction(Z,10,2)

Z = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
0 1 2 3 4 5 6 7 8 9.

simpleFunction(Z, 10, 2)

Inside Function: Y, n, x

total = Y[0] = 1

n = 10

for loopIndex in range(1, n): → 1 : 9

① total = 2 * 1 + Y[1]
= 3

② total = 2 * 3 + Y[2]
= 6 + 1 = 7

③ total = 2 * 7 + Y[3]
= 14 + 1
= 15

④ total = 2 * 15 + Y[4]
= 31

i → $2^{i+1}-1$

① → 3
② → 7
③ → 15
④ → 31

$2^{i+1}-1$

⑨ → $2^{i+1} - 1 = 2^{9+1} - 1$

$= 2^{10} - 1$

total = 1023

```
# GATE 2020

def tob(b: int, arr: list)->int:
  i = 0
  while b>0:
    if b%2:
      arr[i] = 1
    else:
      arr[i] = 0
    b//=2
    i+=1
  return i

def pp(a: int, b: int)->int:
  arr = [None]*20
  ex = a
  tot = 1
  l = tob(b, arr)
  for i in range(l):
    if (arr[i]==1):
      tot = tot*ex
    ex = ex*ex
  return tot

pp(3,4)
```

tob(4, arr)

i = 0

while b > 0:

if b%.2:

arr[i] = 1

else:

arr[i] = 0

b → 4 2 1 0 ; i → 0 1 2 3

arr[0] = 0

arr[1] = 0

arr[2] = 1

i = 3

a b

pp(3,4)

[None, None, None, ... None]
0 1 2 19

ex = a ⇒ ex = 3

tot = 1

l = tob(b, arr) ⇒ 3

arr →	0	0	1	None	None	.....	None
	0	1	2	3	4	.....	19

for i in range(l): 0, 1, 2

if arr[i] == 1: ✓

tot = tot * ex ✓

ex = ex * ex

return tot

ex → 9 → 81 → 6561

81 * 81

tot = 1 * 81 ⇒ 81

62

81

GATE 2023

```
def funcp():
  x = 1
  def innerFunCp():
    nonlocal x
    x += 1
    return x
  return innerFunCp

f = funcp()
x = f()
y = f()+x

print(x+y)
```

f = funcp( )

x = 1 → 2 → 3

f = innerFunCp

x = f( ) ← 2

y = f( ) + x

3 + 2

y = 5

5+2 → 7

2+5 → 7

```
# ISRO 2020

def rer(n: int, r: int)->int:
  if n>0:
    return (n%r+rer(n//r, r))
  else:
    return 0

rer(513, 2)
```

rer(513, 2)

↓

2

513

1000001

rer(513, 2)

↳ 1 + rer(256, 2) = 2

rer(256, 2)

↓

0 + rer(128, 2) = 1

↓

0 + rer(64, 2) = 1

↓

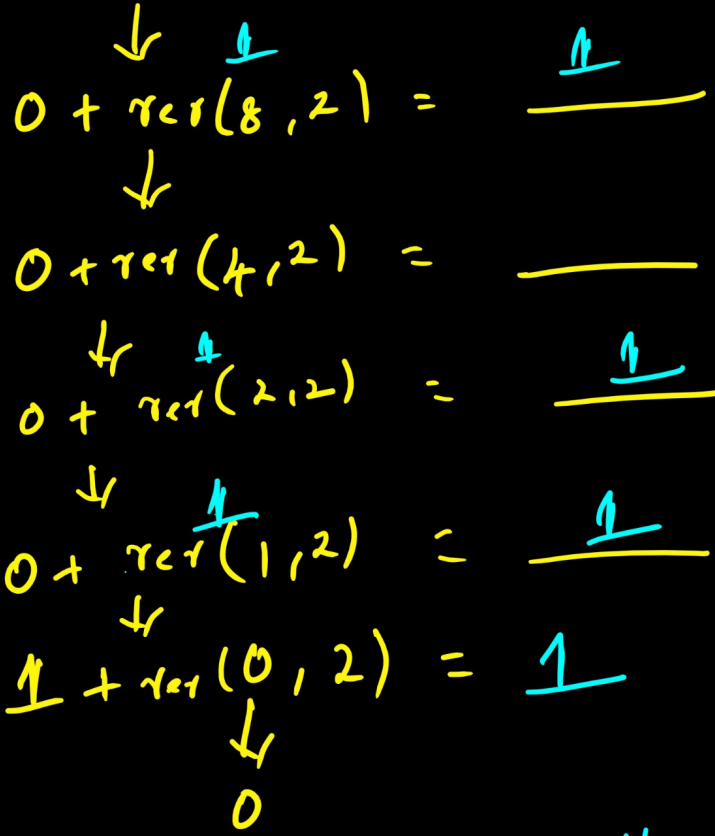
0 + rer(32, 2) = 1

↓

0 + rer(16, 2) = 1

↓

1



GATE 2019

```
def convert(n: int)->None:
  if n<0:
    print(n)
  else:
    convert(n//2)
    print(n)
```

Which of the following will happen when the function convert is called with any positive integer n as argument?

a) It will print the binary representation of n and terminate

b) It will print the binary representation of n in the reverse order and terminate

c) It will print the binary representation of n but will not terminate.

d) None of these. (It will not print anything and will not terminate)

Convert(5)

↓

Convert(2)

↓

Convert(1)

↓

Convert(0)

↓

Convert(0)

```
def jumble(x: int, y: int):
  x = 2*x*y   2*x+y
  return x

x = 2
y = 5
y = jumble(y, x)
x = jumble(y, x)
print(x)
```

a) 13

b) ~~12~~ 26

c) 2

d) 5

x = 5
y = 2
x = 2*5+2
x = ~~10~~ 12
return 12

y = jumble(5, 2)
y = ~~10~~ 12
x = jumble(~~10~~ 12, 2)
x = 26

jumble(12, 2)
x = 12
y = 2
x = 2*12+2
= 26

GATE 2018

```
def Count(x, y):
  if y!=1:
    if x!=1:
      print("*")
      Count(x//2, y)
    else:
      y -= 1
      Count(1024, y)

Count(1024, 1024)
```

The number of times that the print statement is executed by the call Count(1024, 1024) is ——

Count (1024, 1024)

↓

*

Count (512, 1024)

↓

*

Count (256, 1024)

↓

*

Count(1024,1024) →

① * Count(512, 1024)

② * Count(256, 1024)

③ * Count(128, 1024)

④ * Count(64, 1024)

⑤ * Count(32, 1024)

⑥ * Count(16, 1024)

⑦ * Count(8, 1024)

⑧ * Count(4, 1024)

⑨ * Count(2, 1024)

⑩ * Count(1, 1024)

y - - = 1 , y = 1024

y = 1023

Count (1024, 1023)
x y

* Count (512, 1023)
* Count (256, 1027)

⋮

* Count (1, 1023)

y - - = 1

y = 1022

Count (1024, 1022)

1024 → 1023

1023 * 10 = 1023 0

Count (1024, 1024) → 10 * prints

Count (1024, 1023) → 10 * print

⋮

1024, 1022
1024, 1021

⋮

Count (1024, 2) → 10 * prints

Count (1024, 1) → 10 * print

10230

GATE 2017

```
def foo(val: int)->int:
  x = 0
  while val>0:
    x = x+foo(val)
    val -= 1
  return val
```

foo(3) recursion error.

```
def bar(val: int)->int:
  x = 0
  while val>0:
    x = x+bar(val-1)
  return val
```

infinite loop.

Invocations of foo(3) and bar(3) will result in —

a) Return of 6 & 6 respectively

b) Infinite loop and abnormal termination respectively.

c) Abnormal termination and infinite loop respectively

d) Both terminating Abnormally.

foo(3) ⟶ Recursion error.

↳ x=0

val = 3

x = x + foo(val)

↓ 3

foo(3)

↓

x = 0

while val > 0:

x = x + foo(val)

bar(3)

↓

x = 0
val = 3
while val > 0:
    x = x + bar(2)

↓

x = 0
val = 2
while val > 0
    x = x + bar(1)

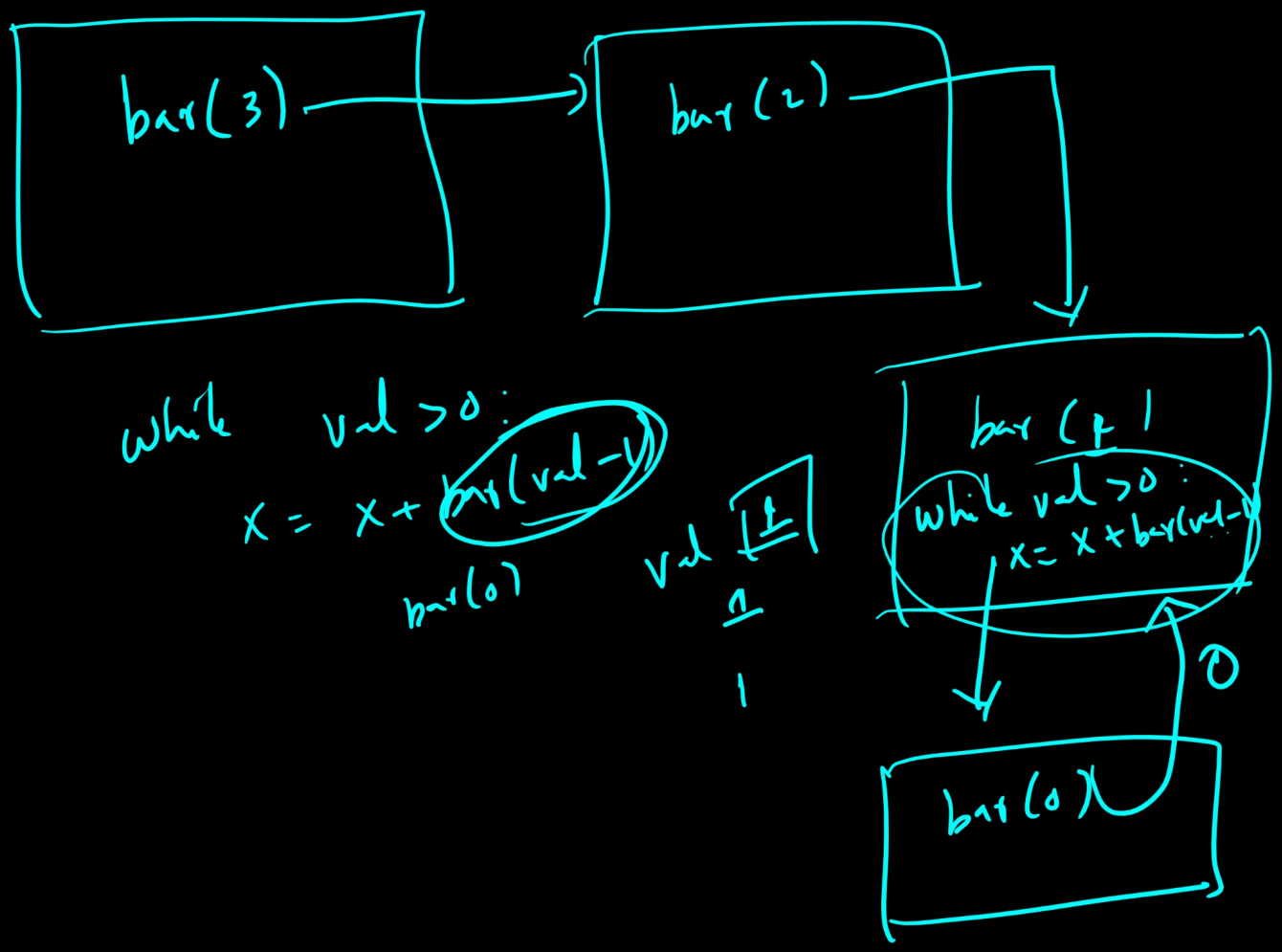
bar(3) → bar(2) → bar(1)

↓

x = 0
val = 1
while val > 0:
    x = x + bar(0)

bar(1) → bar(0) → 0

while val > 0: 0
    x = x + bar(0)
    x = 0




```
# GATE 2017

def fun1(n: int)->None:
  if n==0:
    return
  print(n)
  fun2(n-2)
  print(n)

def fun2(n: int)->None:
  if n==0:
    return
  print(n)
  n+=1
  fun1(n)
  print(n)
```

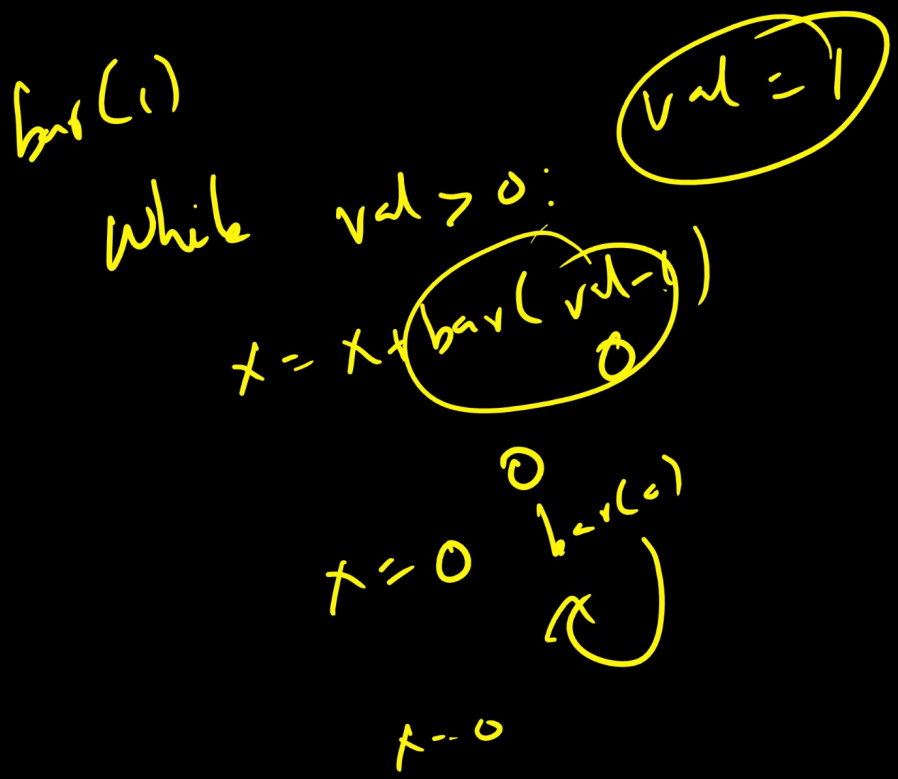
The output printed when fun1(5) is called is —

a) 5 3 4 2 3 1 2 0 1 1 2 2 3 3

b) 5 3 4 2 3 1 2 2 2 3 3 4 4 5

c) 5 3 4 2 3 1 2 2 1 3 2 4 3 5

d) 5 3 4 2 3 1 2 0 2 1 3 2 4 3


bar(1)
while val > 0:
x = x + bar(val-1)
val = 1
0
x = 0
bar(0)
x = 0