WELCOME

to

DSA - 401

Topics:-

DSA - ఫుల్ Dry-run

4 yrs

GATE

1) Recursion ★★★★

S1, S2, S3

501

⇒ DP

2) Backtracking

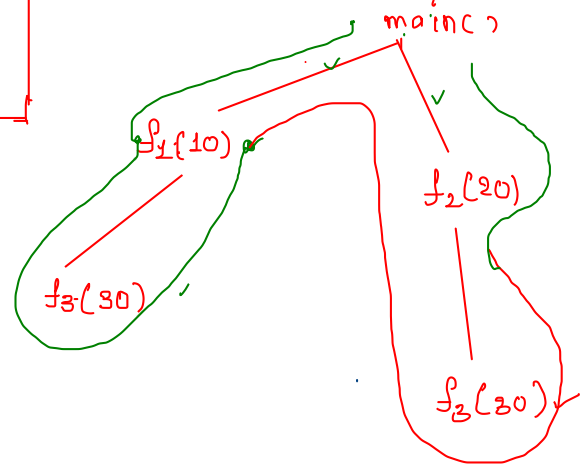
3) Binary Search → S4

10

```
main()
{
    let res1=func1(10)
    let res2=func2(20)
    print(res1)
    print(res1+res2)
}
```

```
function func2(n)
{
    let a=30;
    func3(a)
    print("f2")
    return a;
}
```

```
function func1(n)
{
    let a=30;
    func3(a)
    print("f1")
    return 10;
}
```




```
function func3(n)
{
    let b=30
    n=n+1
    print("f3")
    return a+n;
}
```

o/p	o/p
f_3	f_3
f_1	f_1
f_3	10
f_2	30
10	f_2
40	40

f_3	10
f_1	40
f_3	
f_2	

10

```
main()
{
        let res=fun(4);
        print(res)
}
```

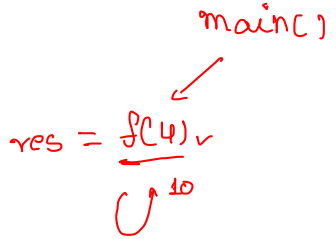
10

```
function fun(n)
{
        let sum=0
        for(let i=1;i<=n;i++)
        {
                sum=sum+i;
        }
        return sum
}
```

4

4 times sum=0 1 2 3 4

Loop's



```
function fun(n)
{
        let sum=0
        for(let i=1;i<=n;i++)
        {
                sum=sum+i;
        }
        return sum
}
```

```
main()
{
        let res=fun(4)
        print(res)
}
```

f(4): 1+2+3+4

4+3+2+1

```
function fun(n)
{
        return n + fun(n-1);
}
```

f(4) → 4 + f(3) → 3 + f(2) → 2 + f(1) → 1 + f(0) → 0 + f(-1) → -1 + f(-2)

```
function fun(n)
{
        if(n==0) return 0
        return n + fun(n-1);
}
```

ret f(4) → 4 + f(3) → 3 + f(2) → 2 + f(1) → 1 + f(0) → 0

n = 0 ✓ valid input

```
function fun(n)
{
    if(n==1) return 1
      return n + fun(n-1);
}
```

→ Terminating / base-condn

condn

f(0)
↓
ret 0 + f(-1)
↓
ret -1 + f(-2)
⋮

Loops:-

```
function fun(n)
{
        let sum=0
        for(let i=1;i<=n;i++)
        {
                sum=sum+i;
        }
        return sum
}
```

function calling (Recursion)

```
function fun(n)
{
        if(n==0) return 0
        return n + fun(n-1);
}
```

f(4)
↳ f(3) ✓
↳ f(2) ✓
↳ f(1) ✓
↳ f(0)
✓ Terminated

4 + 3 + 2 + 1

```
main()
{
        let res=fun(4)
        print(res)
}
```

Introduction to Recursion – Data Structure and Algorithm Tutorials

Difficulty Level : Easy • Last Updated : 26 Feb, 2023

What is Recursion?

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a recursive function. Using a recursive algorithm, certain problems can be solved quite easily. Examples of such problems are Towers of Hanoi (TOH), Inorder/Preorder/Postorder Tree Traversals, DFS of Graph, etc. A recursive function solves a particular problem by calling a copy of itself and solving smaller subproblems of the original problems. Many more recursive calls can be generated as and when required. It is essential to know that we should provide a certain case in order to terminate this recursion process. So we can say that every time the function calls itself with a simpler version of the original problem.

Need of Recursion

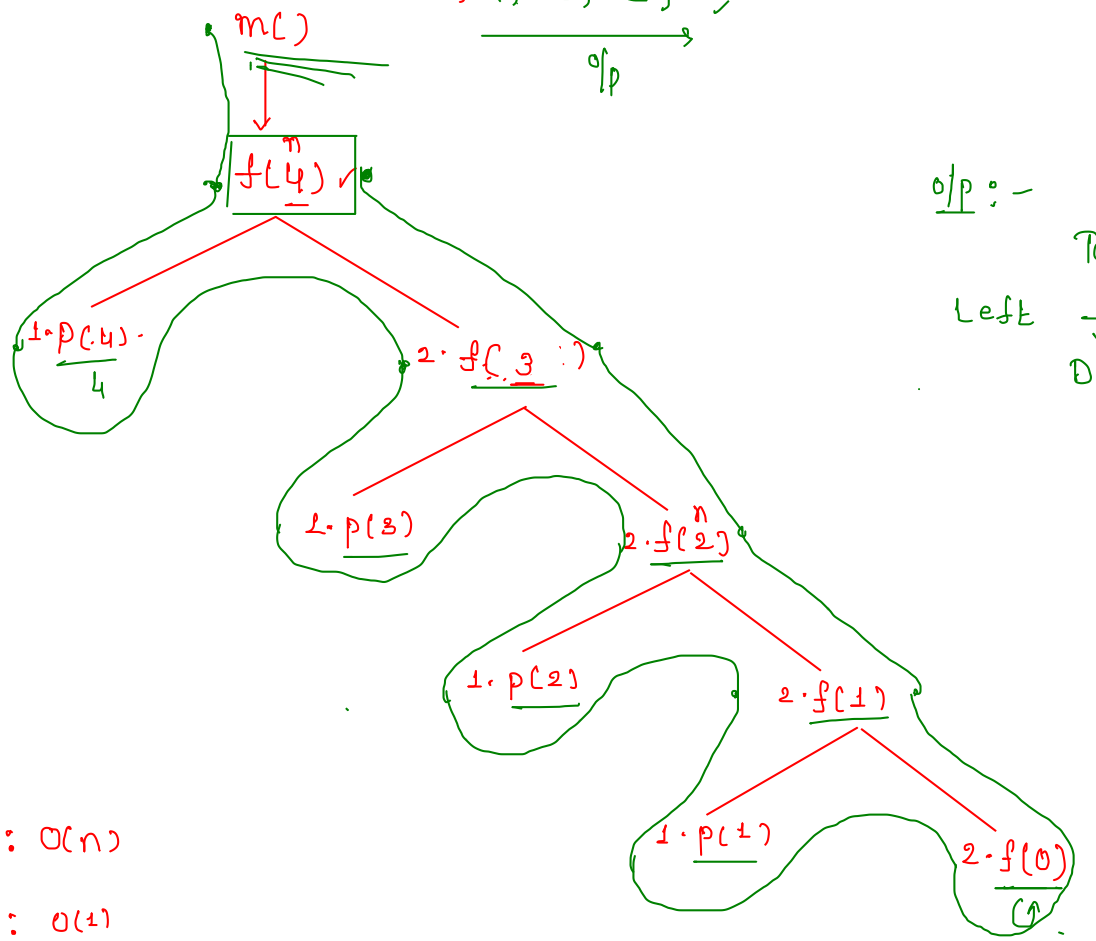
Recursion is an amazing technique with the help of which we can reduce the length of our code and make it easier to read and write. It has certain advantages over the iteration technique which will be discussed later. A task that can be defined with its similar subtask, recursion is one of the best solutions for it. For example; The Factorial of a number.

Properties of Recursion:

```
fun(n)
{
  if(n==0)
      return ;
  else
    {
       1. print(n)
       2. fun(n-1)
    }
}
main()
{
     fun(4)
}
```

Recursion Tree Op

4, 3, 2, 1



O/p :-

Top

Left → Right

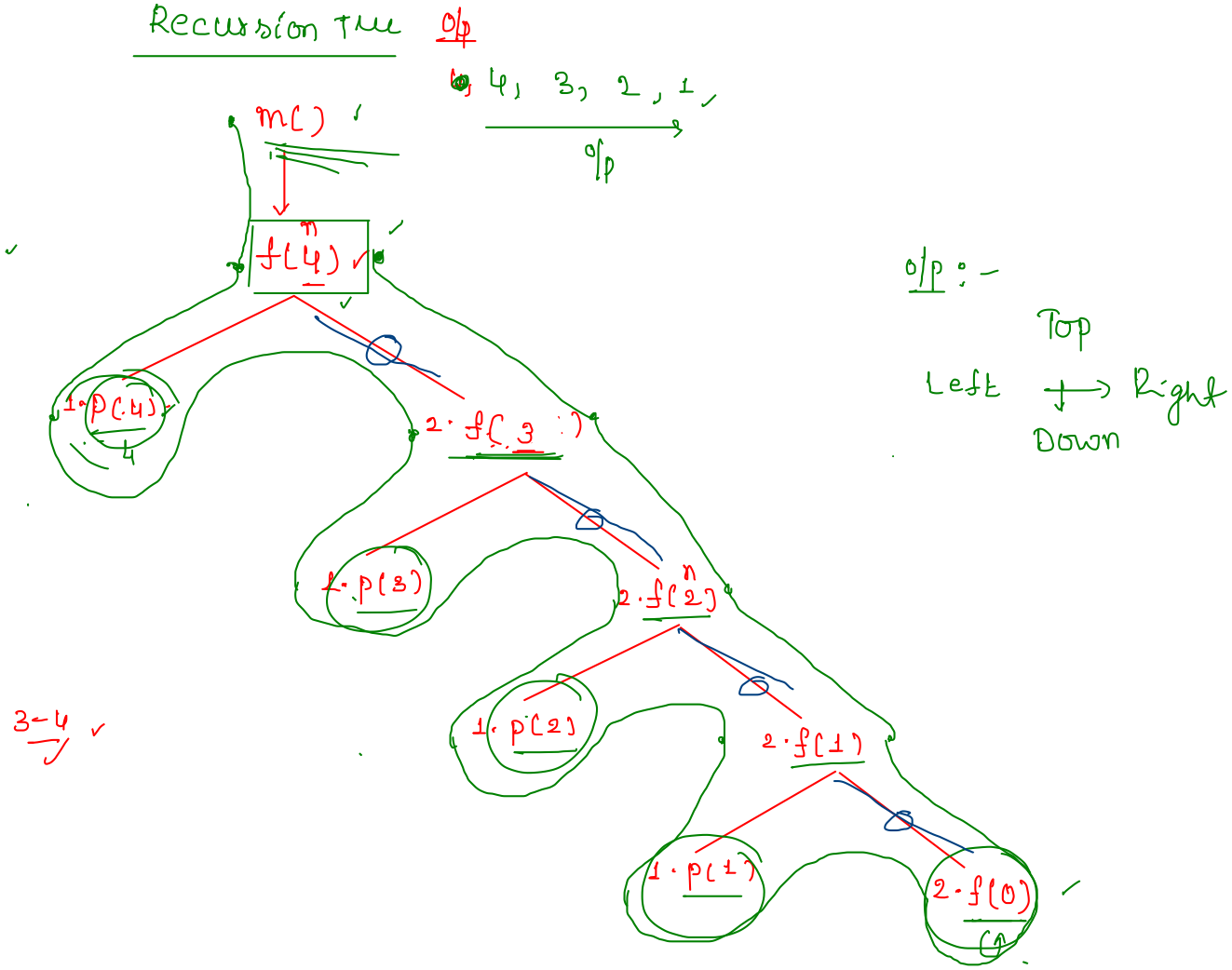
Down

```
function fun(n)
{
    for(let i=n; i>=1; i--)
    {
        print(i)
    }
}
```

T.C : O(n)

S.C : O(1)

```
fun(n)
{
  if(n==0)
      return ;
  else
    {
       print(n)
       fun(n-1)
    }
}
main()
{
     fun(4)
}
```



Analysis of T.C and S.C in recursion programgs

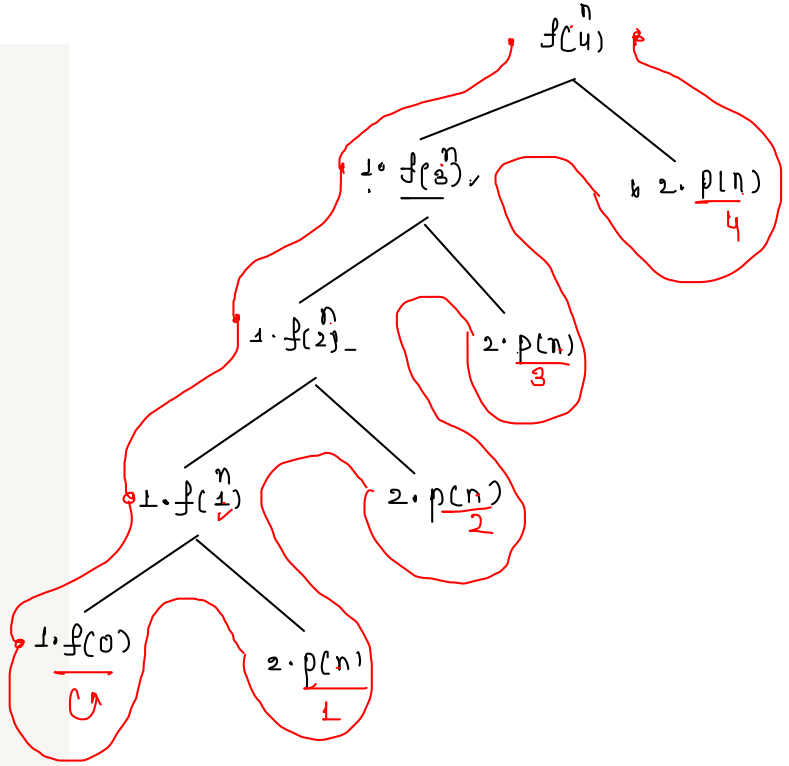
1) T.C :- Draw the recursion tree, count the number of function class, compare with given n value

2) S.C :- Draw the recursion tree, count number of edges starting from root node to longest leaf node

↳ No - more children

```
fun(n)
{
   if(n==0)
      return
   else
   {
    1. fun(n-1)
    2. print(n)

   }
}
main()
{
     fun(4)
}
```



op

1, 2, 3, 4

f(0) : No o/p

f(1) : 1

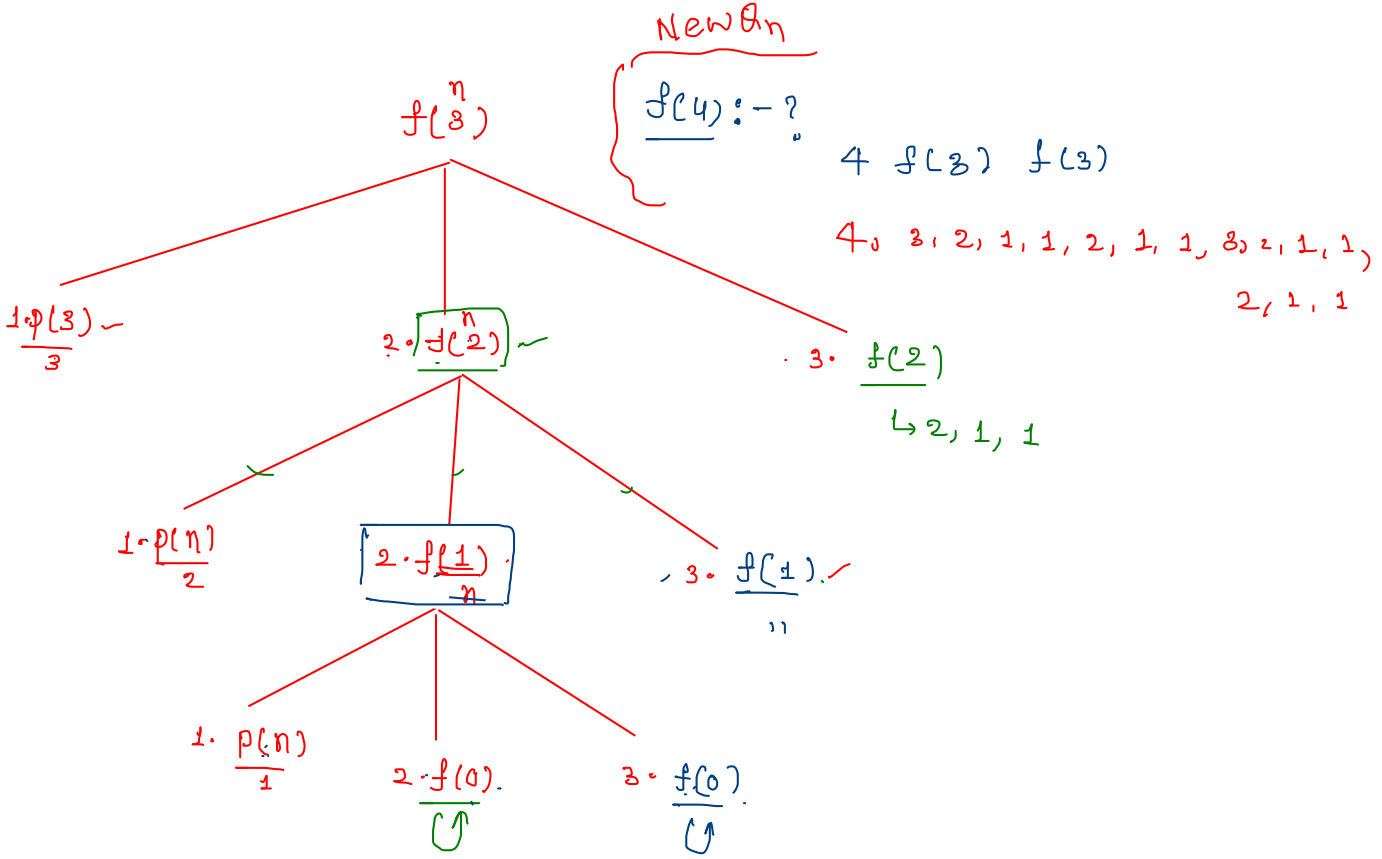
f(2) : 2, 1, 1

o/p :- 3, 2, 1, 1, 2, 1, 1

f(2) → f(1) f(1)

```
fun(n)
{
   if(n==0)
      return
   else
   {
       1. print(n)
       2. fun(n-1)
       3. fun(n-1)
    }
}
main()
{
    fun(3)
}
```

fun(4)



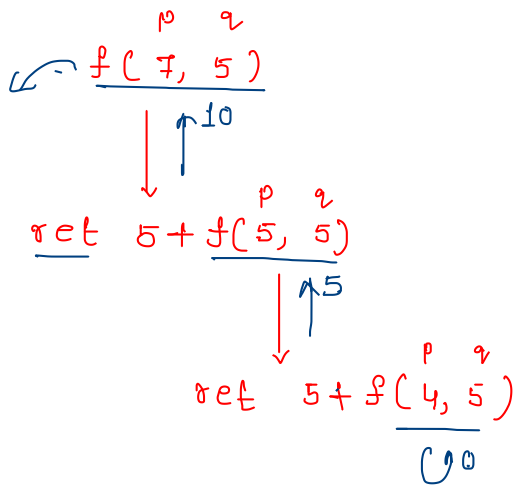
4 f(3) f(3)

4, 3, 2, 1, 1, 2, 1, 1, 3, 2, 1, 1, 2, 1, 1

[10, 12, 16]

2-3 mins

```
int fun(int p,int q)
{
   if(p<q)
        return 0
    else if(p==q)
        return p+fun(p-1,q)
    else
        return q+fun(p-2,q)
}
main()
{
     print(fun(7,5))
}
```



GATE CSE 2023 | Question: 26

asked in Programming Feb 15 · retagged Feb 20 by **Lakshman Patel RJIT**

2

Q.26 Consider the following program:

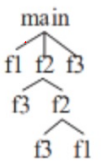
```
int main()
{
   f1();
   f2(2);
   f3();
   return(0);
}
```

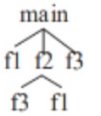
```
int f1()
{
  return(1);
}
```

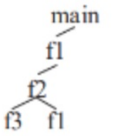
```
int f2(int X)
{
   f3();
  if (X==1)
    return f1();
  else
    return (X*f2(X-1));
}
```

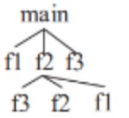
```
int f3()
{
  return(5);
}
```

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

(A) 

(B) 

(C) 

(D) 

GATE CSE 2023 | Question: 26

asked in Programming Feb 15 · retagged Feb 20 by Lakshman Patel RJIT

2

Q.26 Consider the following program:

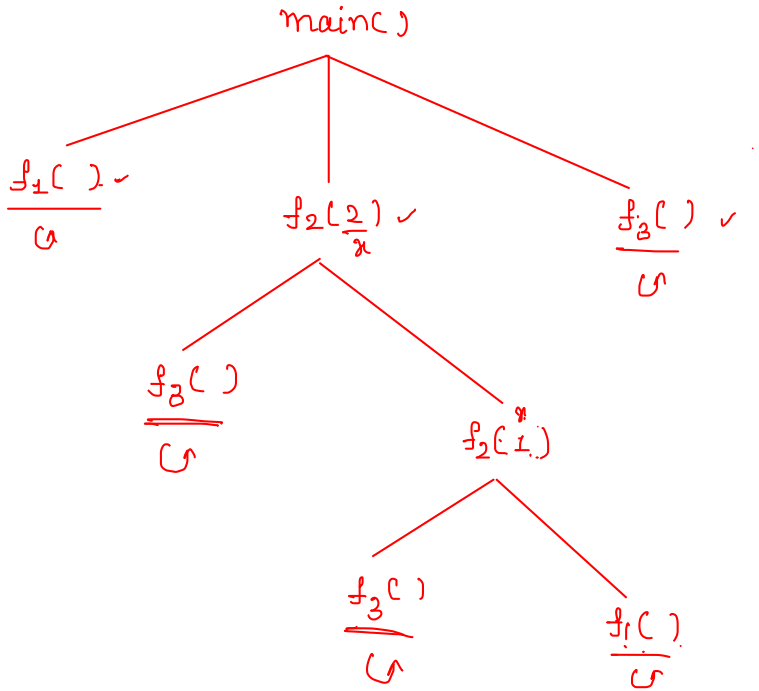
```
int main()
{
  f1();
  f2(2);
  f3();
  return(0);
}
```

```
int f1()
{
 return(1);
}
```

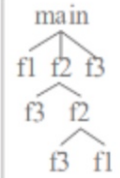
```
int f2(int X)
{
  f3();
  if (X==1)
    return f1();
  else
    return (X*f2(X-1));
}
```

```
int f3()
{
  return(5);
}
```

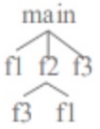
Which one of the following options represents the activation tree corresponding to the main function?



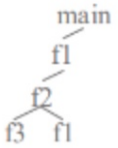
(A) ✓


main
f1 f2 f3
f3 f2
f3 f1

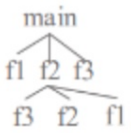
(B)

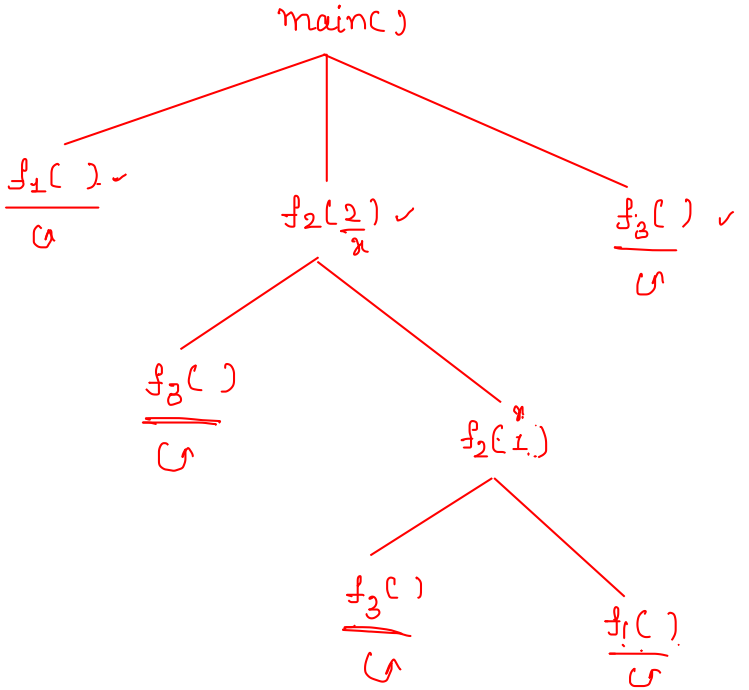

main
f1 f2 f3
f3 f1

(C)


main
f1
f2
f3 f1

(D)


main
f1 f2 f3
f3 f2 f1


main()
f1( ) ✓
f2(2) ✓
f3( ) ✓
f3( )
f2(1)
f3( )
f1( )

GATE CSE 2011 | Question: 48

asked in **Algorithms** Sep 29, 2014 • edited Jun 13, 2018 by **Milicevic3306**

17

Consider the following recursive C function that takes two arguments.

```
unsigned int foo(unsigned int n, unsigned int r) {
    if (n>0) return ((n%r) + foo(n/r, r));
    else return 0;
}
```

What is the return value of the function foo when it is called as $\text{foo}(345, 10)$?

A. 345
B. 12
C. 5
D. 3

GATE CSE 2011 | Question: 49

asked in **Algorithms** Apr 21, 2016 • edited Nov 2, 2017 by **kenzou**

17

Consider the following recursive C function that takes two arguments.

```
unsigned int foo(unsigned int n, unsigned int r) {
    if (n>0) return ((n%r) + foo(n/r, r));
    else return 0;
}
```

What is the return value of the function foo when it is called as $\text{foo}(513, 2)$?

A. 9
B. 8
C. 5
D. 2

GATE CSE 2018 | Question: 45

asked in **Algorithms** Feb 14, 2018 • retagged Dec 1, 2022 by **Lakshman Patel RJIT**

34

Consider the following program written in pseudo-code. Assume that x and y are integers.

```
Count (x, y) {
    if (y !=1 ) {
        if (x !=1) {
            print("*");
            Count (x/2, y);
        }
        else {
            y=y-1;
            Count (1024, y);
        }
    }
}
```

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

GATE CSE 2017 Set 1 | Question: 36

asked in **Programming** Feb 14, 2017 • edited Jan 23, 2018 by **kenzou**

93

Consider the C functions foo and bar given below:

```
int foo(int val) {
    int x=0;
    while(val > 0) {
        x = x + foo(val--);
    }
    return val;
}
```

```
int bar(int val) {
    int x = 0;
    while(val > 0) {
        x= x + bar(val-1);
    }
    return val;
}
```

Invocations of $foo(3)$ and $bar(3)$ will result in:

A. Return of 6 and 6 respectively.
B. Infinite loop and abnormal termination respectively.
C. Abnormal termination and infinite loop respectively.
D. Both terminating abnormally.

GATE CSE 2017 Set 1 | Question: 35

asked in **Programming** Feb 14, 2017 • edited Jan 27, 2018 by **kenzou**

40

Consider the following two functions.

```
void fun1(int n) {
    if(n == 0) return;
    printf("%d", n);
    fun2(n - 2);
    printf("%d", n);
}
void fun2(int n) {
    if(n == 0) return;
    printf("%d", n);
    fun1(++n);
    printf("%d", n);
}
```

The output printed when $\mathrm{fun1}(5)$ is called is

A. 53423122233445
B. 53423120112233
C. 53423122132435
D. 53423120213243

GATE IT 2008 | Question: 82

asked in **Algorithms** Oct 29, 2014 • edited Jun 25, 2018 by **Pooja Khatri**

14

Consider the code fragment written in C below :

```
void f (int n)
{
  if (n <=1)  {
   printf ("%d", n);
  }
  else {
   f (n/2);
   printf ("%d", n%2);
  }
}
```

What does f(173) print?

A. 010110101
B. 010101101
C. 10110101
D. 10101101