C-107 ⇒ **Types of Recursion**

Tail and Non Tail Recursion

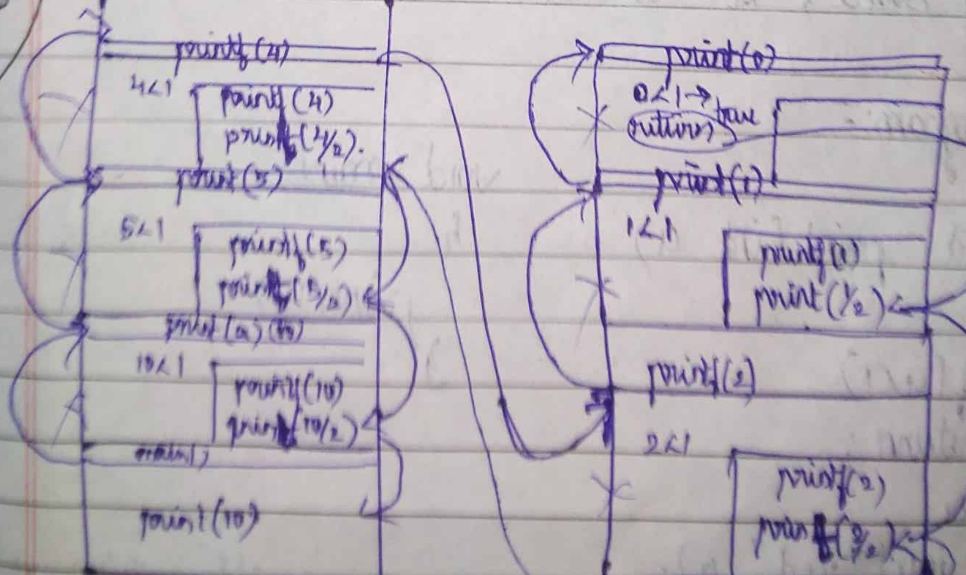
* Last thing to do in function or there would be no task left after the execution of recursive calls is called as Tail Recursion.

Program: (How to find tail or non tail recursion)

```
void print(int a)
{
 if (a<1)
 return;
 else
  printf("%d", a);
  print(a/2);
}
void main()
{
 print(10);
}
```

This is single pgm and by looking we can guess it's tail recursion

after this recursive call there is no statement but we can't tell it tail recursion by just only seeing it.



O/P ⇒ 10 5 4 2 1

* After the recursive call there is no call to be executed here, and here it is tail recursion.

* Important thing is, we cannot say by seeing a program it is a tail recursion; instead we should work with the flow of execution of code to find it is tail or non tail recursion.

* Tail recursive programs are like loop; so better to write these programs in loops because in functions we maintain a stack of memory in which tail recursion would consume waste of memory or memory space is simply occupied without usage of memory.

* But time complexity for loops and recursion tail function is $O(n)$. or it is same.

* But space complexity for both is not same, it is $O(1)$

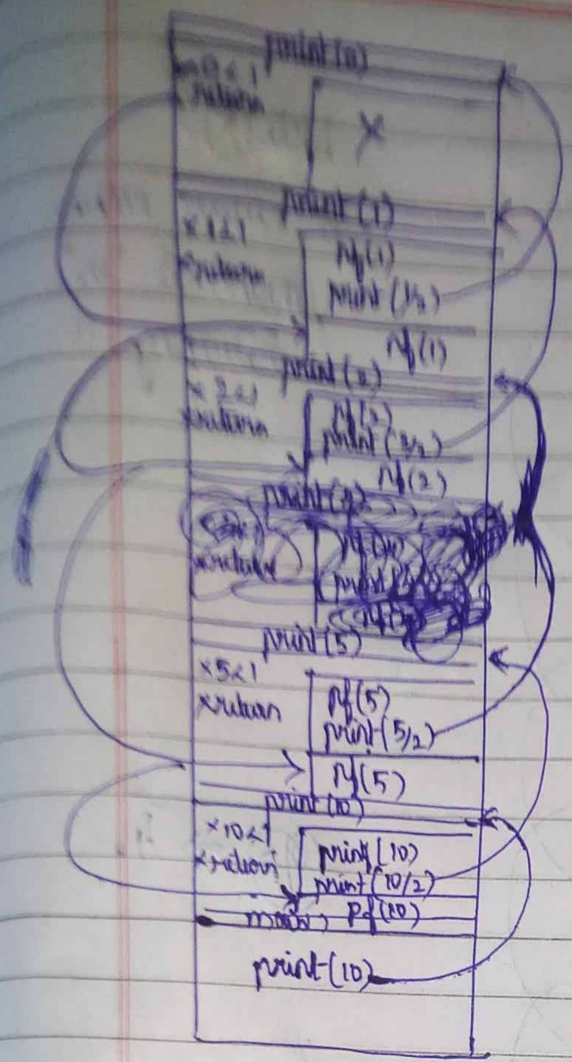
Program:-

```
void print(int a)
{
if(a<1)
return;
else
printf("%d", a);
print(a/2);
}
```

```
void main()
{
print(10);
}
```

This is simple program and we guess it is non tail

after recursive call we have statement and we guess it is non tail



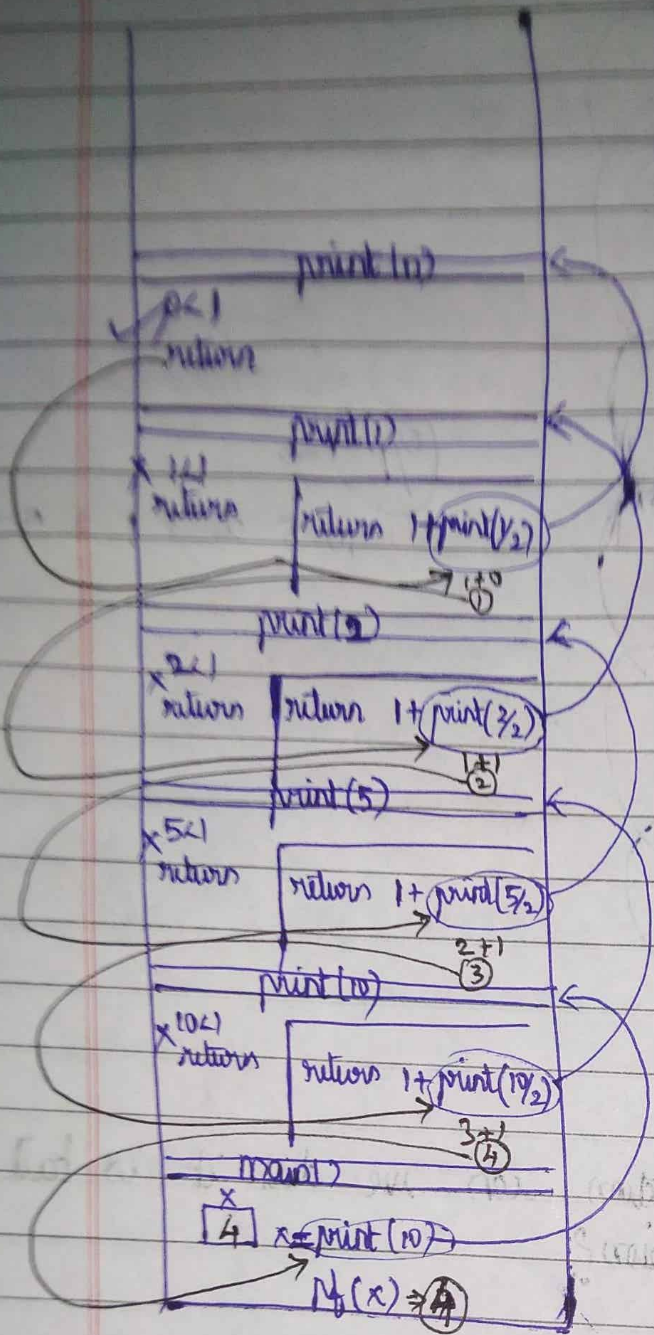
O/P ⇒ 10 5 2 1 1 2 5 10.

Program:—
By looking this function can we tell it is tail or non tail recursion?

```
void print(int a)
{
 if (a<1)
 return;
else
return 1+ print(a/2);
}
void main()
{
  int x;
  x = print(10);
  printf ("%d", x);
}
```

This is non tail recursion but by looking at it we guess tail; so check the flow

Note: By looking at this we guess it is tail recursion but it is non tail recursion so check the flow of execution.



Note (X)

* return nothing means it is returning 0.

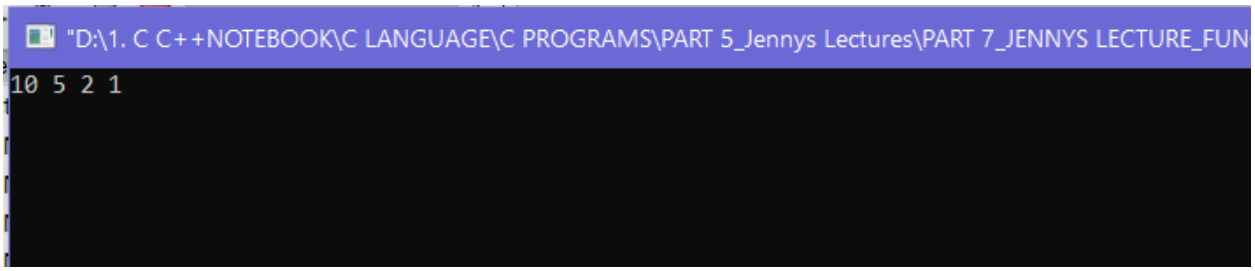
O/P ⇒ 4.

* So this is non tail recursion; and here we fully utilize the stack memory till the entire life time of program.

But tail recursion; will not utilize the memory and hence it leads to wastage of memory.

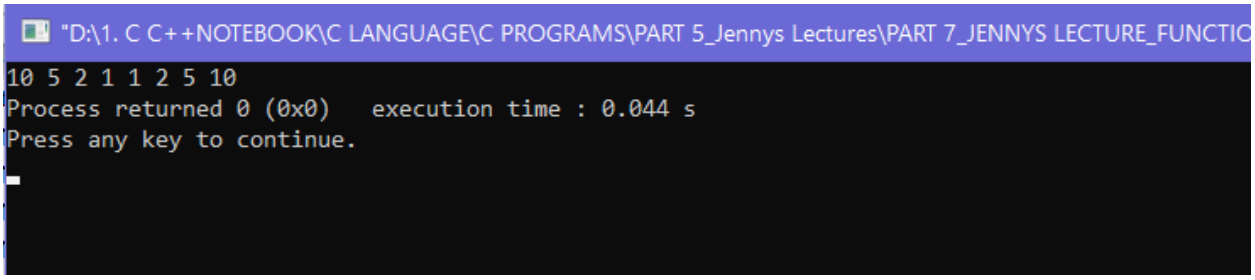
```c
#include <stdio.h>
#include <stdlib.h>
/** 1-TAIL RECURSION **/

void print(int a)
{
 if(a<1)
  return;
 else
  printf("%d ",a);
 print(a/2);
}
int main()
{
    print(10);
    getch();
}
```



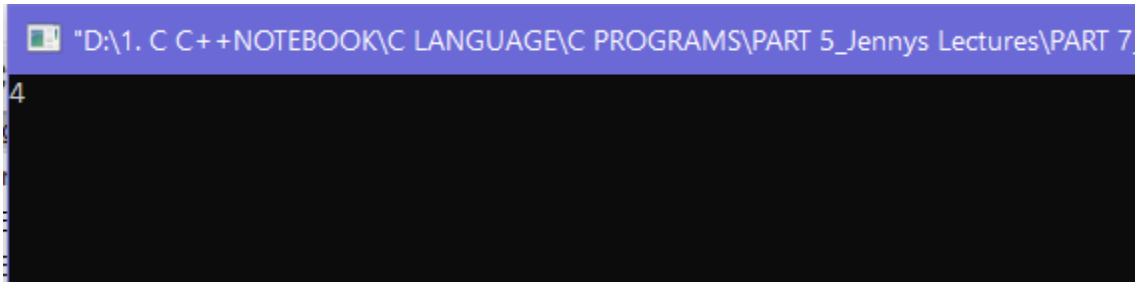

```c
#include <stdio.h>
#include <stdlib.h>
/** 2-NON TAIL RECURSION **/

void print(int a)
{
 if(a<1)
  return;
 else
  printf("%d ",a);
 print(a/2);
 printf("%d ",a);
}
int main()
{
    print(10);
}
```

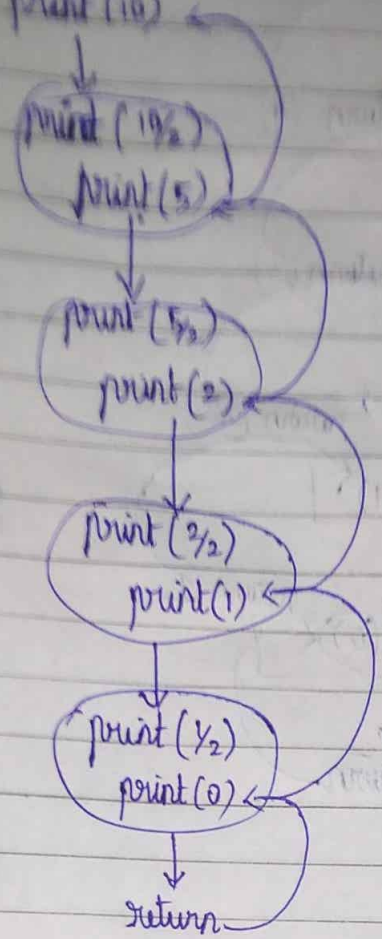



```c
#include <stdio.h>
#include <stdlib.h>
/** 3-NON TAIL RECURSION **/

int print(int a)          //print function should have return type int not void
{
 if(a<1)
  return;
 else
  return 1+print(a/2);
}
int main()
{
    int s;
    s=print(10);
    printf("%d",s);
    getch();
}
```

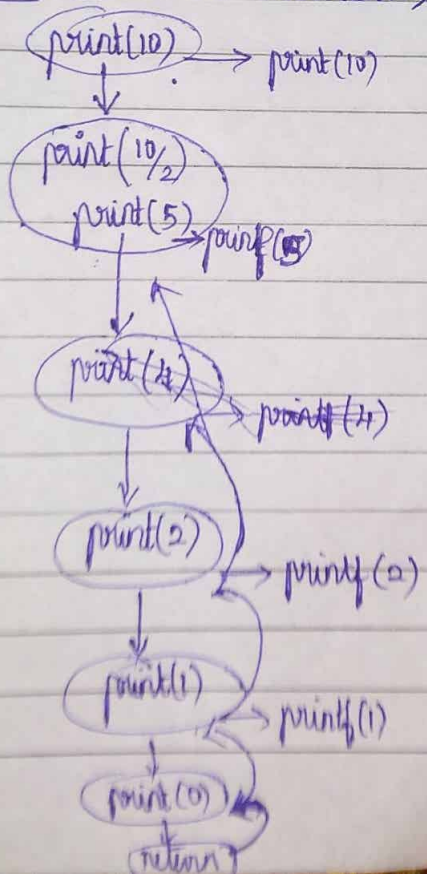


Program (Tail Recursion)



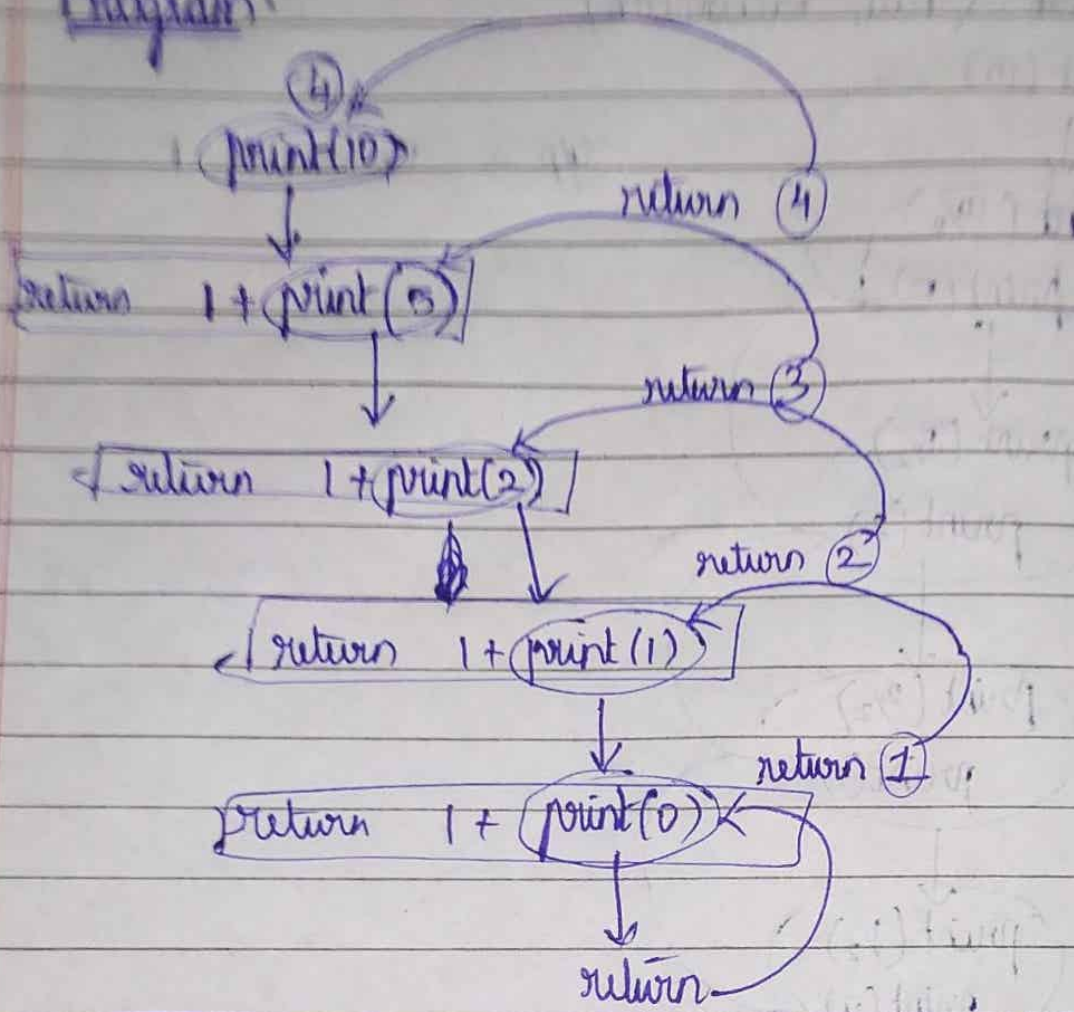
O/P ⇒ 10 5 2 1

Program (Non Tail Recursion)



O/P ⇒ 10 5 2 1 1 2 5 10

Diagram:


4
print(10)
return 1 + print(3)
return 1 + print(2)
return 1 + print(1)
return 1 + print(0)
return
return 1
return 2
return 3
return 4

O/P → 4