C_104 ⇒ Recursion in C

Introduction to Recursion

Recursion → when a function call itself [illegible] directly or indirectly then it is recursion.

```
Eg:-
        void add()
        {
            ----
            -----
            -----
            add(); → Recursion
            ---
            ---
        }
```

* Function can call itself by any ways either directly or indirectly, based on that we have types of recursion.

* ~~Example;~~ assume teacher one function and another function jerry, when teacher calls jerry at some point jerry responds to teacher and process over.

* But, what happens when jerry calls itself, there is no other jerry; so this process leads to infinite loop,

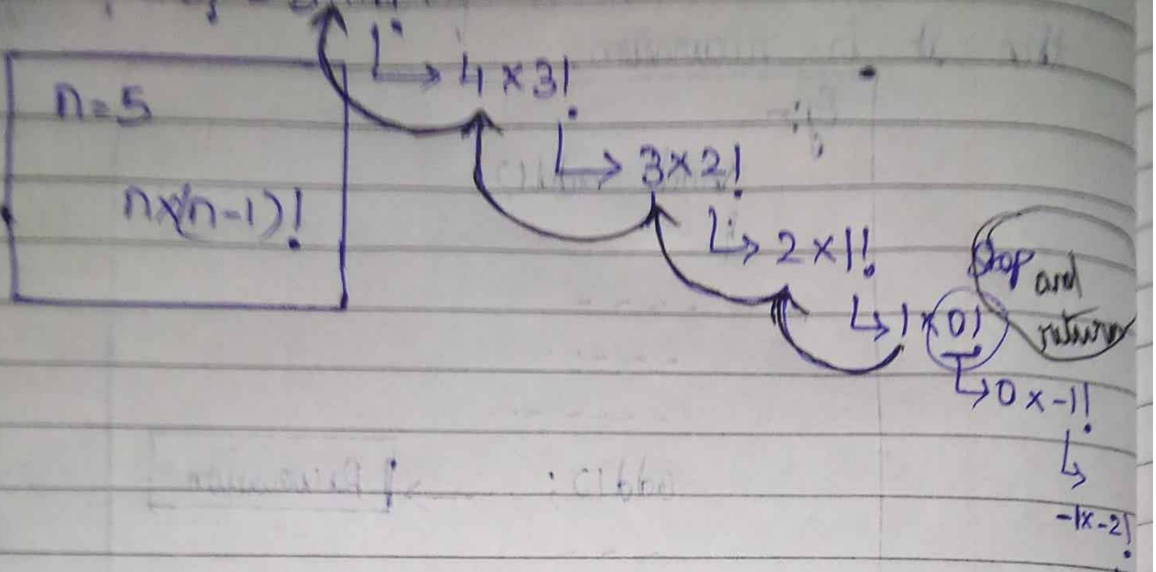
* So when the recursion should stop.

* This is the base condition or terminate condition.

* so we have to put base condition very correctly at proper place to terminate loop.

5! = 5×4×3×2×1

5! = 5×4!



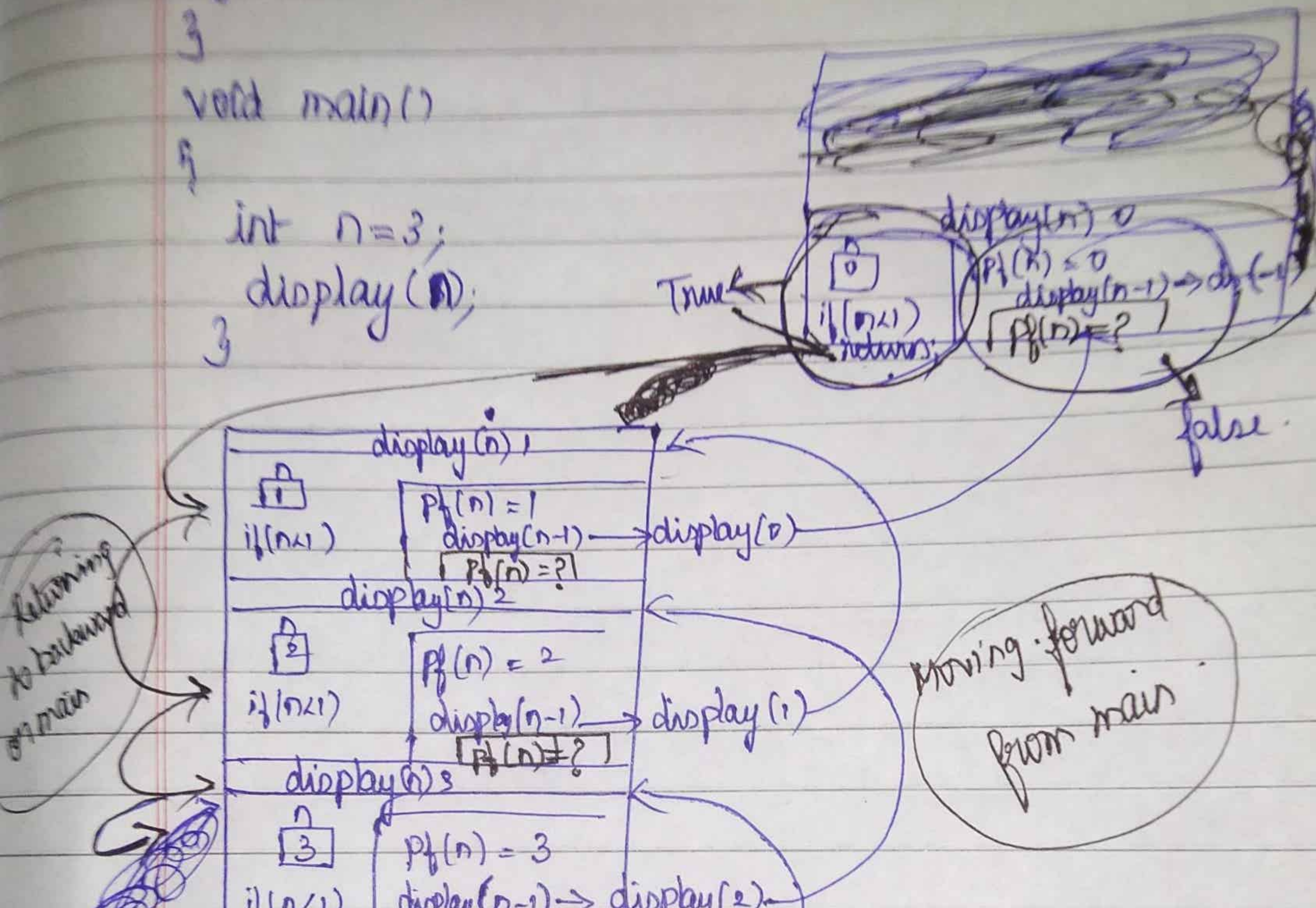
* so the condition goes on, so stop the loop with condition (n==0) or (n<=1).

* when n==10; there is no chance to terminate the loop; in this case there will be infinite loop and it leads to stack overflow problem.

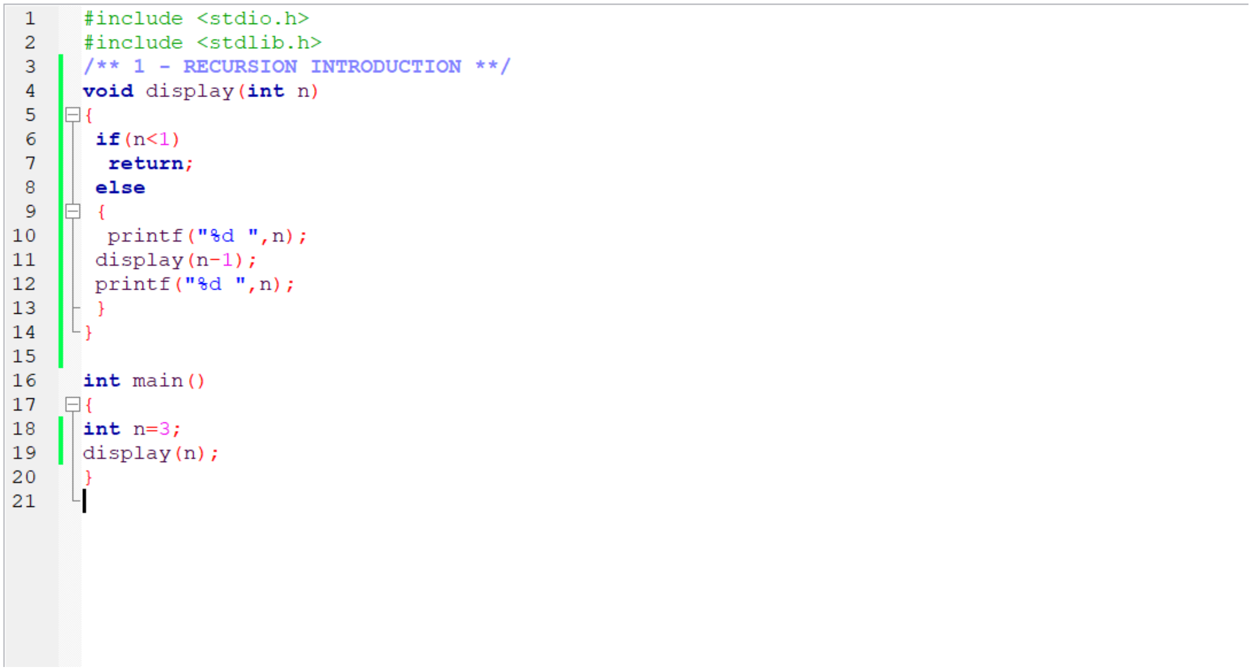
Program:

```
#include<stdio.h>
void display(int n)
{
 if(n<1)
 return;
 else
 {
   printf("%d", n);
```

```
    display(n-1);
    printf("%d",n);
  }
}
void main()
{
  int n=3;
  display(n);
}
```

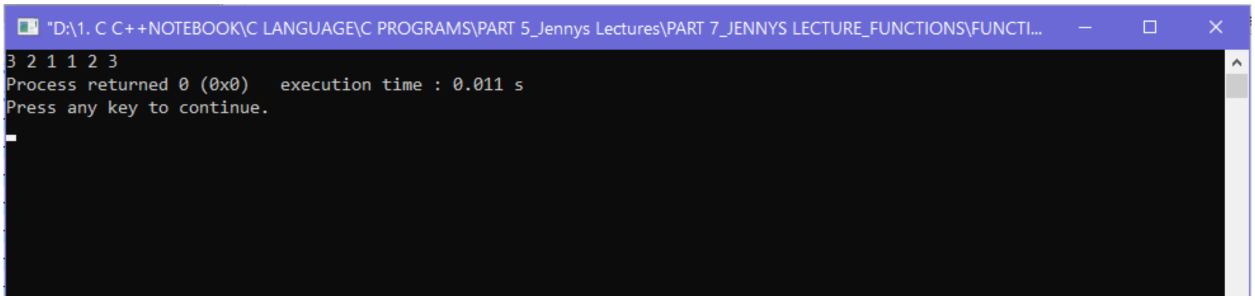


o/p => 3 2 1 1 2 3



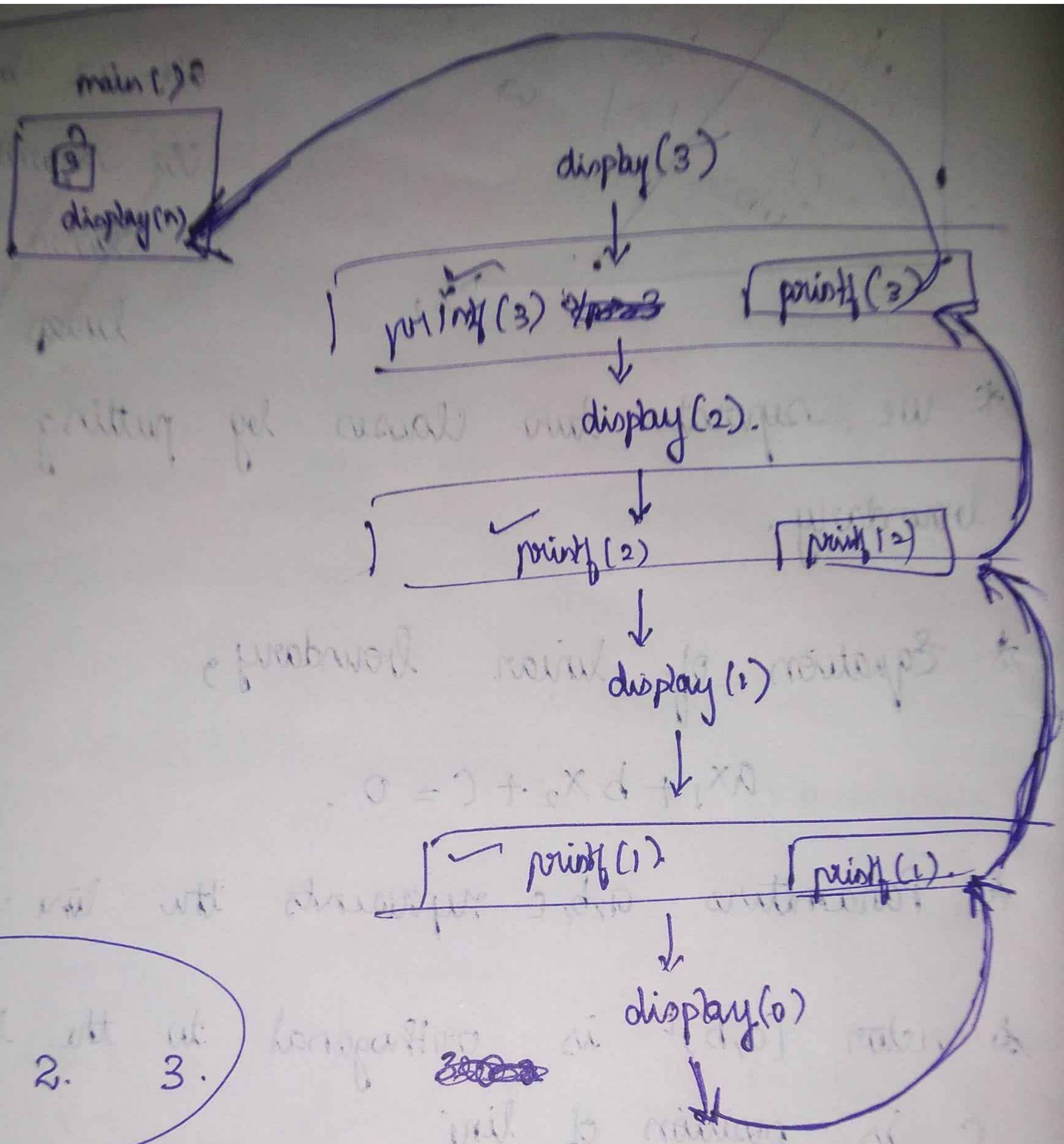
```c
#include <stdio.h>
#include <stdlib.h>
/** 1 - RECURSION INTRODUCTION **/
void display(int n)
{
 if(n<1)
  return;
 else
 {
  printf("%d ",n);
 display(n-1);
 printf("%d ",n);
 }
}

int main()
{
int n=3;
display(n);
}
```



"D:\1. C C++NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 7_JENNYS LECTURE_FUNCTIONS\FUNCTI...

```
3 2 1 1 2 3
Process returned 0 (0x0)   execution time : 0.011 s
Press any key to continue.
```


main(){
3
display(n)
display(3)
printf(3)
printf(3)
display(2).
printf(2)
printf(2)
display(1)
printf(1)
printf(1).
display(0)

3 2 1 1 2. 3.