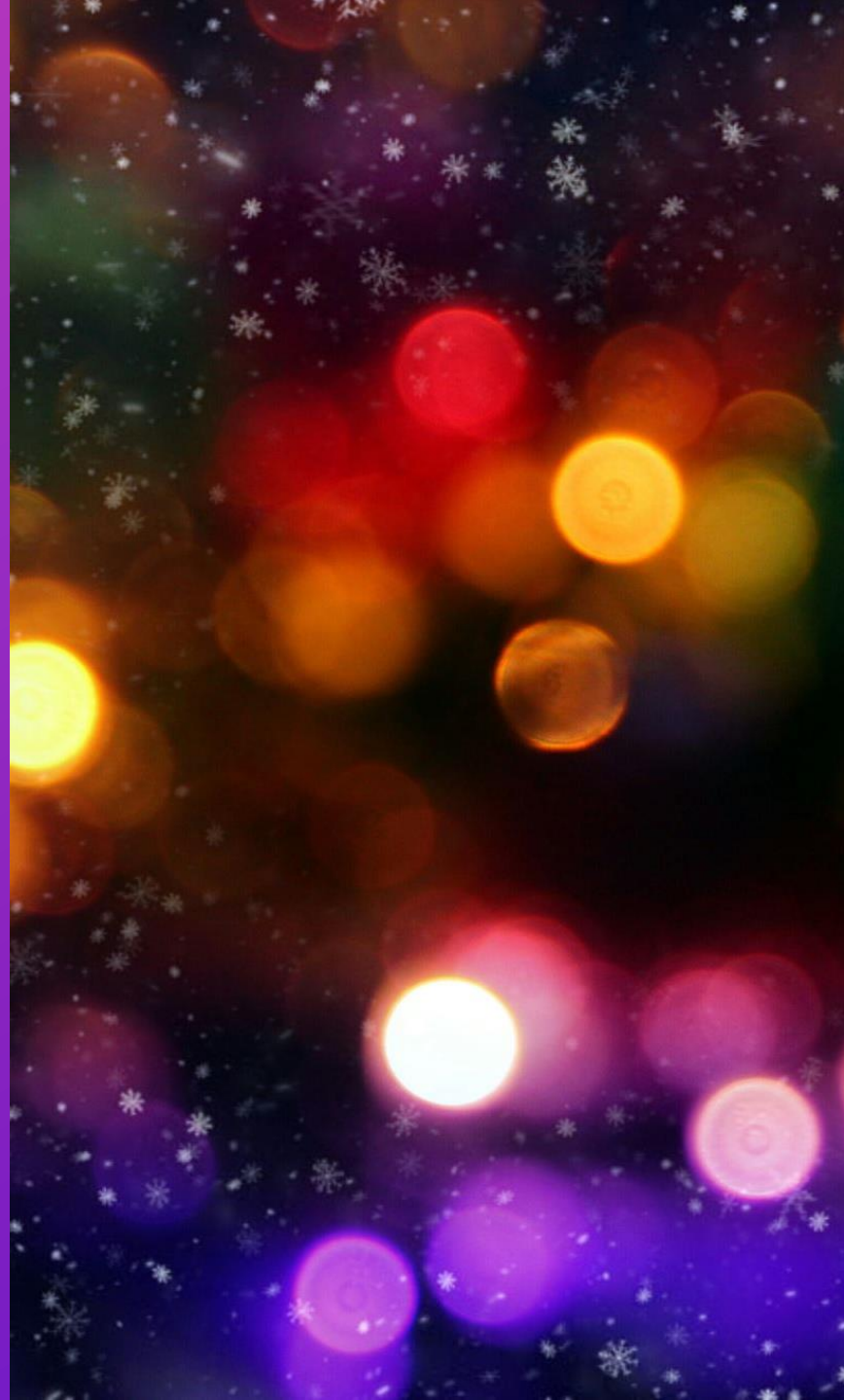
MNNIT COMPUTER CODING CLUB

CLASS-7

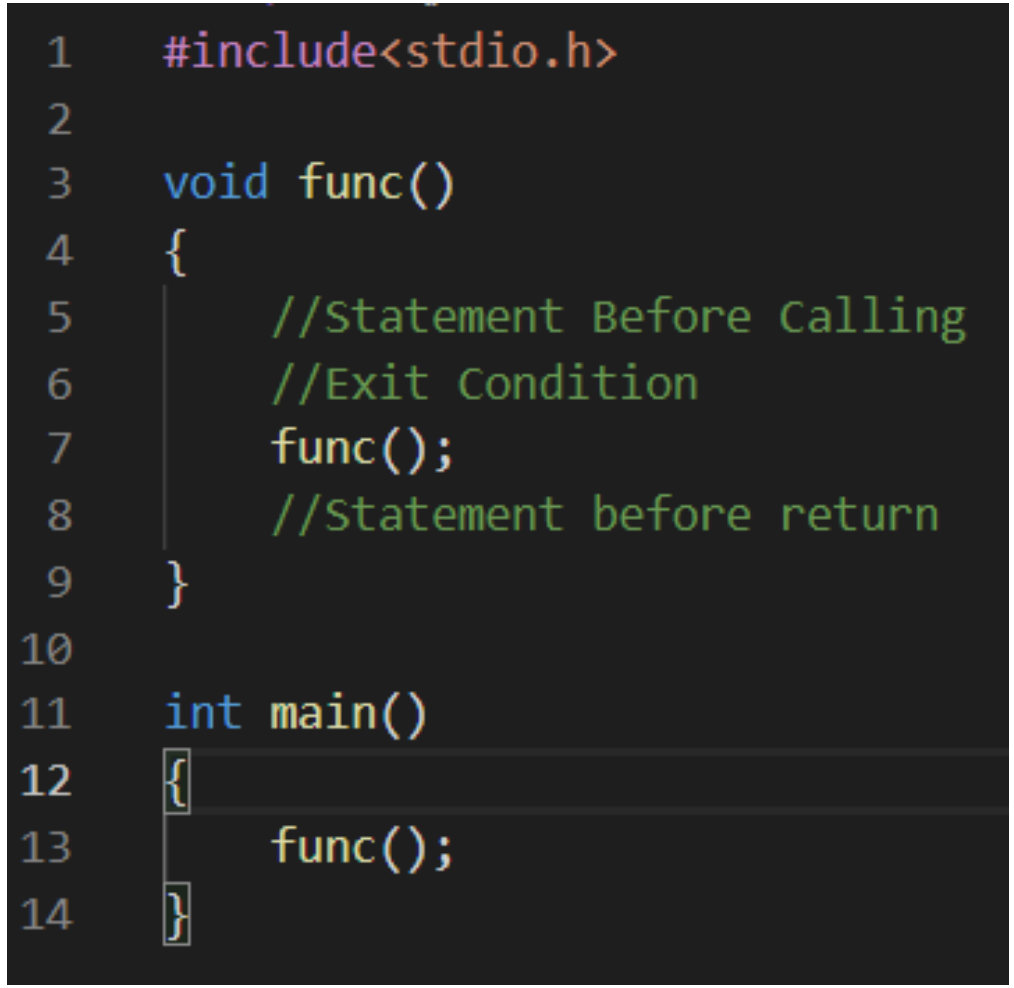
BASICS OF C



RECURSION

- A technique where a function calls itself in loop.
- A powerful construct to solve complex problems easily

- Before writing a recursive function for a problem we should consider these points:
  - We should be able to define the solution of the problem in terms of a similar type of smaller problem.
  - At each step we get closer to the final solution of our original solution
  - There should be a terminating condition to stop recursion.



```c
#include<stdio.h>

void func()
{
    //Statement Before Calling
    //Exit Condition
    func();
    //Statement before return
}

int main()
{
    func();
}
```

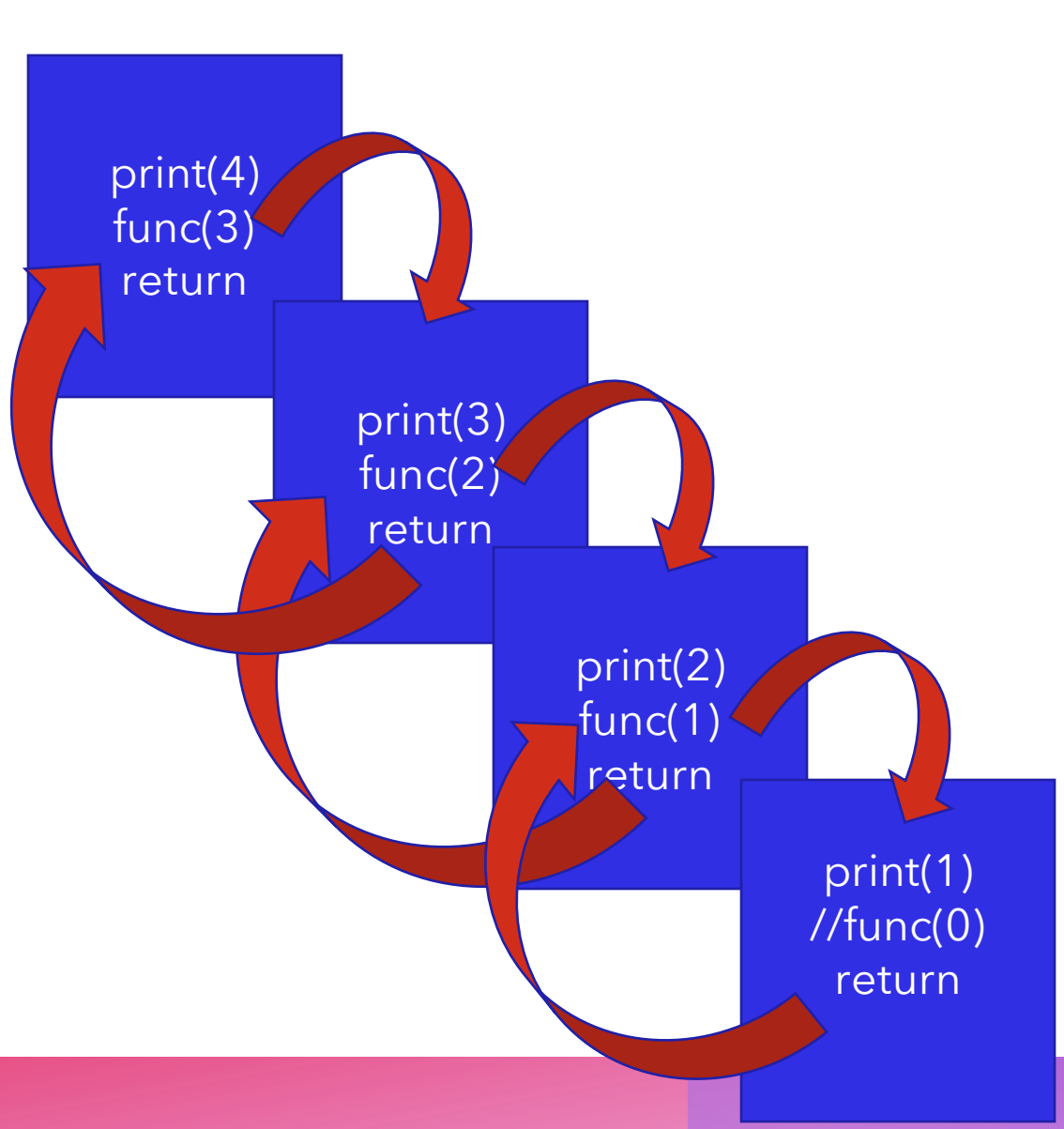
PROBLEM: PRINT 1 TO 10

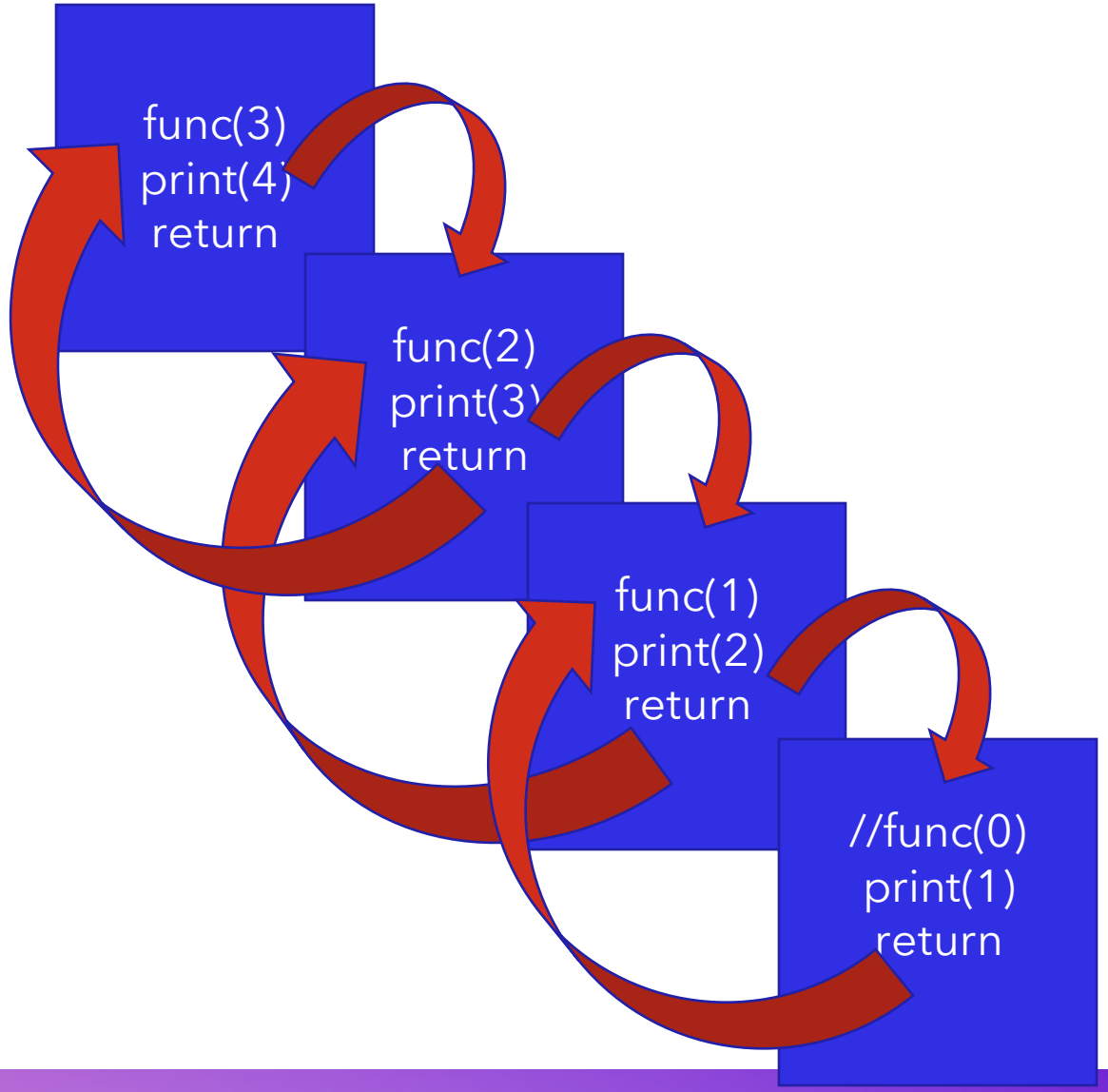
- Here we need to give 2 variants of this solution. One where numbers are printed in ascending order and other with descending order.
- Step 1: Determine the subproblem
- Step 2: Identify Exit Condition
- Step 3: Determine where to solve the current iteration (before recursive call or after recursive call)

```
3   void printNumbers(int n)
4   {
5       if(n>1)
6           printNumbers(n-1);
7       printf("%d\n", n);
8   }
```

```
3   void printNumbers(int n)
4   {
5       printf("%d\n", n);
6       if(n>1)
7           printNumbers(n-1);
8   }
```

HOW IS IT WORKING...??





FACTORIAL OF A NUMBER

```c
#include<stdio.h>
int main()
{
    int n;
    scanf("%d", &n);
    int fact = 1;
    for(int i=1; i<=n; i++)
    {
        fact *= i;
    }
    printf("%d\n", fact);
}
```

```c
#include<stdio.h>
int fact(int n)
{
    if(n == 1)
        return 1;
    int sol = n * fact(n-1);
    return sol;
}
int main()
{
    int n;
    scanf("%d", &n);
    printf("%d\n", fact(n));
}
```