→ **Recursion:** It is a process in which a function calls itself directly or indirectly is called recursion

① **sum of N numbers:**

```
int sumofN(int n)
{
    if(n==1)      } Base case
      return 1;   }
    return n + sumofN(n-1);
}
```

② calculate n^P (n power p)

```
int power(int n, int p)
{
    if(p==0)      } Base case.
      return 1;   }
    return n * power(n,p-1);
}
```

③ **factorial of n:**

```
int factorial (int n)
{
    if (n==0)          } Base case
        return 1;      }
    return n*factorial(n-1);
}
```

④ <u>fibonacci at n:</u>

```
int fibo(int n)
{
    if (n<=1)          } Base case.
        return n;      }
    return fibo(n-1)+fibo(n-2);
}
```

⑤ check if an array is sorted:

```
bool isSorted(int Arr[], int n)
{
    if (n==1)              } Base case
        return true;       }
    return (Arr[0] < Arr[1] && isSorted
                                (Arr[]+1,n-1);
}
```

(6) print numbers till N (increasing order):

```
void printNincreasing (int n)
{
    if (n==0)    } Base case
      return;    }
    printNincreasing (n-1);
    cout<< n;
}.
```

(7) print numers till N (decreasing order):

```
void printNDecreasing (int n)
{
    if (n==0)    } Base case
      return;    }
    cout << n;
return printNDecreasing (n-1);
}.
```

(8) first occurence of an element in an Array:

```
int firstOccurence (int Arr[], n, i, K)
{
    if (~~[illegible]~~ i == n)
        return -1;
    if (A[i] == K)
        return i;
    return firstoccurence (Arr, n, i+1, K);
}
```

(9) <u>Last occurence:</u>

```
int lastOccurence (int Arr[], int n, int i,
                                int K)
{
    if (i == n)
        return -1;
    int restArray = lastoccurence (Arr, n, i+1, K)
    if (restArray != -1)
        return restArray;
    if (Arr[i] == K)
        return i;
    return -1;
}
```