Rekursion

```
void function(input size)
{
    base case ⟶ must be defined and reached!
    .. .. ...
    function(smaller input size) //recursive call
    ... ...
}

int main()
{
    ... .. ...
    function(input size)
    ... .. ...
}
```

⟶ smaller subproblems must eventually reach the base case

⟶ code must cover all smaller subproblems

⟶ base case cannot have recursive calls i.e. defined in a non-recursive manner.

⟶ recursive call should call with smaller input size and should make progress toward base case

⟶ in each call: think how you can use the solution of smaller subproblems to solve the current input size

⟶ you can have several different calls here dep on some condition / constraint on the input e.g. n odd ⟶
n even ⟶

code examples: power function

```
3   // POST: result == x^n
4   unsigned int power(const unsigned int x, const unsigned int n)
5   {
6       if (n == 0)
7       {
8           return 1;
9       }
10      else if (n == 1)
11      {
12          return x;
13      }
14      else if (n % 2 == 0)
15      {
16          int temp = power(x, n / 2);
17          return temp * temp;
18      }
19      else
20      {
21          return x * power(x, n - 1);
22      }
23  }
```

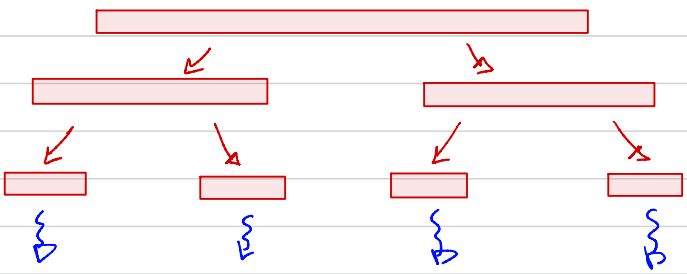
two non-recursive calls

two recursive calls

⟶ this shows nicely, that it is possible and sometimes smart to differentiate between several base cases and several recursive calls.

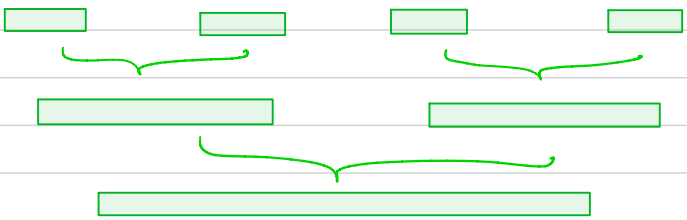
"Divide and Conquer"

e.g. problem of size n



divide problem into smaller subproblems

conquer the smaller subproblems ≙ solve problem for small size



combine the results from the smaller problems to get final result

```cpp
#include<iostream>

unsigned int partial_sum(const unsigned int n) {
  if (n == 0)
    return 0;
  else {
    // print descending
    // std::cout << n << std::endl;
    unsigned int partial = partial_sum(n - 1);
    // print ascending
    std::cout << n << std::endl;
    return n + partial;
  }
}

int main() {
  std::cout << "n = ";
  unsigned int n;
  std::cin >> n;
  std::cout << partial_sum(n) << std::endl;
  return 0;
}
```

PS(5)