```
1
2
3   Recursive ‘Algorithms’{
4
5     [LAPC CS 131]
6
7
8
9        return   Presentation
10
11
12  }
13
14
```

By Romeo Azizian

# What is Recursion {

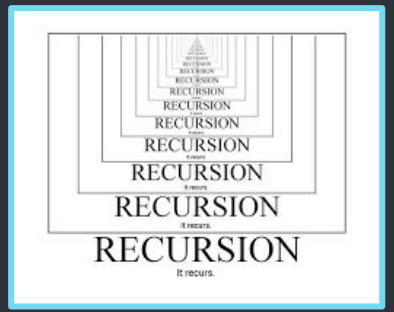
In computer science,
<An algorithm is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input.
>

< /1 >

- *Direct*
- a call to a function appears in that function's body

< /2 >

- *Indirect*
- the pattern is some function calls some other function

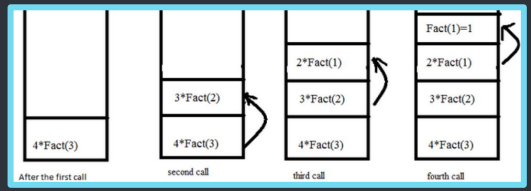



}

# Iteration vs Recursion {

| RECURSION | Iteration |
| --- | --- |
| Uses more storage space requirement | Less storage space requirement |
| Overhead during runtime | Less Overhead during runtime |
| Runs slower | Runs faster |
| a better choice, a more elegant solution for recursive problems | Less elegant solution for recursive problems |

}




Call stack

# Real World Problem {

Problem: collect $1,000.00 from a 1000 people in the room by asking them for **1 dollar.**

*Iterative solution*: visit the 1,000 people, and ask each for a $1

*Recursive solution*: give a $1 to person otherwise visit 10 people and ask them each to collect 1/10 the amount that you are asked to raise; collect the money they give you into one bag; give this bag to the person who asked you for the money

}

# Form: Directly Recursive Function {

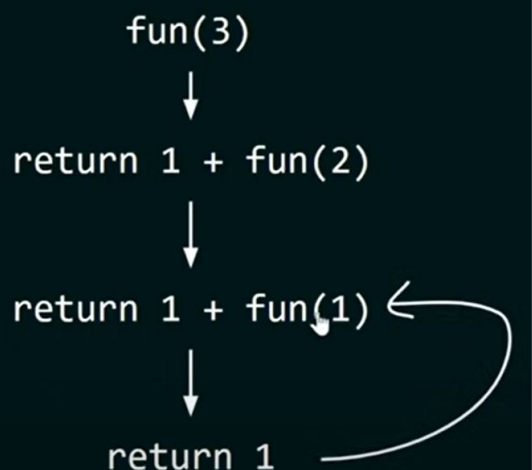
```
def Solve(Problem):
  if Base case then Solve Problem directly and return solution
  else:
    (1) Decompose Problem
    (2) Recursively call Solve (this function) on each smaller subproblem
    (3) Combine the returned solutions to smaller subproblems
    (4) Return the solution to the Solve Function
```

}

# Ex: Factorial {

```
int fun(int n)
{
    if(n == 1)
        return 1;
    else
        return 1 + fun(n-1)
}

int main() {
    int n = 3;
    printf("%d", fun(n));
    return 0;
}
```




}

# Proving Recursive Functions {

```
1
2
3   procedure square(n: nonnegative integer)
4   if n = 0 then return 0
4   else return square (n - 1) + 2(n - 1) + 1
5
6
7   Using Mathematical Induction:
8   Base Case n=0:
9     0^2 = 0, then that's satisfied
10  Inductive Case:
11    Assume k is an int
11    Then for k + 1:
12            square(k) + 2(k+1) - 1 = k^2 + 2k + 1 = (k + 1)^2
13
14  }
```

# Cited Works {


Dale, W. (2018, May 31). *Recursion: The Pros and Cons - William Dale*. Medium. https://medium.com/@williambdale/recursion-the-pros-and-cons-76d32d75973a#:%7E:text=Recursion%20uses%20more%20memory.,that%20of%20an%20iterative%20function.

*ICS-33*. (2022). Richard Pattis. https://www.ics.uci.edu/~pattis/ICS-33/

Lab, M. (2019, July 18). *Difference between recursion and iteration*. Microcontrollers Lab. https://microcontrollerslab.com/difference-between-recursion-and-iteration/

*Recursion in C*. (2018, December 12). YouTube. https://www.youtube.com/watch?v=kepBmgvWNDw

Rosen, H. K., & Hussain, S. (2022). *Solution Manual of Discrete Mathematics and its Application Kenneth H. Rosen : Students Solution Guide Expert Book | 7th Addition* (7th ed.). Kenneth Rosen.


}