**Recursion**

THEORY :

Function calls itself until a specific condition is met.

1. Base Condition: stop condition is the base condition.
2. Stack Overflow: occurs for the non-executed functions.

The solution to a problem depends on the same nature of the problem on a smaller input size.

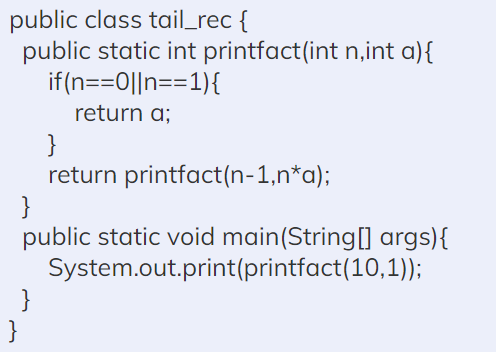
A segmentation fault occurs when you try to access large memory that the system does not have

Call on a smaller size problem, use the output of the smaller size problem and then use that output and find your output and return it. Make sure that t stops somewhere.

Consider you want to climb stairs and you are standing on the floor that is near the first step and someone taught you to climb the first stair and then using that you can climb as many stairs as you want to climb

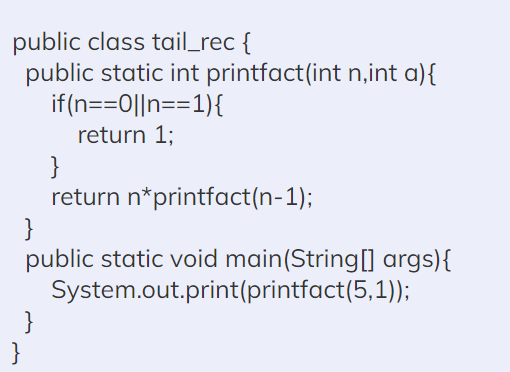
TAIL RECURSION VS NORMAL RECURSION

1. A recursive function is called Tail recursive if the recursive call is the last thing done by the function. There is no need to keep records of the previous state. The tail [recursion](https://www.codingninjas.com/codestudio/library/recursion) uses the recursive function as the last statement of the function. So in it, nothing is left to do after coming back from the recursive call
2. TAIL RECURSION:



Hereafter the base condition encounter, we just return the value of a, nothing to do after that.

1. NORMAL RECURSION



Hereafter the base condition, we return the value 1, then we use the recursion stack to calculate the value of the factorial of 5.

1. TAIL VS NORMAL

The tail recursion is better than Normal recursion. There is no task left after the recursive call; it can be bypassed, more manageable for the compiler to optimize the code. When one function is called, its address is stored inside the stack. So if it is tail recursion, then storing addresses into the stack is not needed.

DISADVANTAGES OF RECURSION

Programs made with recursion have massive storage requirements as compared to when made with an iterative approach. This is due to all the functions remaining in the stack till the termination point has been reached.  
Recursion takes massive amounts of stack space when it is a complex program. This uses more memory and the functions continue to add every call and store its value till the call is executed. When working on certain common problems, recursion can also prove to be slower as compared to the iterative approach.

ADVANTAGES OF RECURSION

Learning recursion well can lead to much more compact, elegant, and optimized codes when building programs. And, by adopting a recursive approach, programmers can save a lot of time on many common real-world, statistical, functional, operational, and mathematical problems.

IF THE RECURSIVE APPROACH IS ALWAYS SLOWER AND CONSUMES MEMORY SHOULD ONE USE IT?

The recursive method is often slower and consumes stack memory on each consecutive function call compared to an iterative approach. It shines when you need to code faster and write smaller functions that are less prone to bugs, such as the case with a tower of Hanoi problem that would otherwise be difficult to solve without recursion. check the iterative approach of tower of hanoi its difficult to understand.

**COMBINATION QS**

1. Print all subsets(array)

<https://www.codingninjas.com/codestudio/problems/find-all-subsets_2041970>

<https://leetcode.com/problems/subsets/submissions/>

1. Print all subsets(string)

<https://practice.geeksforgeeks.org/problems/power-set4302/1>

1. Print all permutations

<https://leetcode.com/submissions/detail/716330865/>

1. Permutation with spaces

<https://practice.geeksforgeeks.org/problems/permutation-with-spaces3627/1>

1. Print all possible combinations of r elements in a given array of size n

<https://leetcode.com/problems/combinations/submissions/>

1. Print all increasing sequences of length k from the first n natural numbers

<https://leetcode.com/problems/combinations/submissions/>

1. Print sums of all subsets of a given set

https://www.codingninjas.com/codestudio/problems/subset-sum\_3843086?topList=striver-sde-sheet-problems&utm\_source=striver&utm\_medium=website

1. Count ways to express a number as a sum of powers

https://www.geeksforgeeks.org/count-ways-express-number-sum-powers/

1. Print all combinations of factors
2. Print all subsequences of a string
3. Write a program to print all permutations of a given string
4. Find the Maximum number possible by doing at-most K swaps
5. Leetcode qs

<https://leetcode.com/problems/combinations/submissions/>

1. <https://www.codingninjas.com/codestudio/library/sum-of-the-combination-of-numbers-part-1>
2. https://www.codingninjas.com/codestudio/library/sum-of-the-combination-of-numbers-part-2
3. Letter case permutation

<https://leetcode.com/problems/letter-case-permutation/submissions/>

1. Generate all the balanced parenthesis

<https://leetcode.com/problems/generate-parentheses/>

1. Different ways to add parentheses

<https://www.codingninjas.com/codestudio/library/different-ways-to-add-parentheses>

1. Print N-bit binary numbers having more 1s than 0s

<https://practice.geeksforgeeks.org/problems/print-n-bit-binary-numbers-having-more-1s-than-0s0252/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article>

1. All possible binary numbers of length n with an equal sum in both halves
2. Generate all binary strings without consecutive 1’s

<https://www.codingninjas.com/codestudio/problems/binary-strings-with-no-consecutive-1s_893001>

1. Print encoding

<https://ide.geeksforgeeks.org/244b84da-145f-4034-8855-dd97dde153bb>

1. Print paths

[https://www.pepcoding.com/resources/online-java-foundation/recursion-with-arraylists/get-stair-paths-official/ojquestion#](https://www.pepcoding.com/resources/online-java-foundation/recursion-with-arraylists/get-stair-paths-official/ojquestion)

1. Return keypad

<https://leetcode.com/problems/letter-combinations-of-a-phone-number/>

1. Flood fill

<https://leetcode.com/problems/flood-fill/>

1. Maximum Score Words Formed by Letters

<https://leetcode.com/problems/maximum-score-words-formed-by-letters/>

1. Game of execution
2. Lexicographical numbers

https://leetcode.com/problems/lexicographical-numbers/

**OTHER QS**

1. Kth symbol in the grammar

<https://leetcode.com/problems/k-th-symbol-in-grammar/>

1. Tower of Hanoi

<https://www.codingninjas.com/codestudio/problems/tower-of-hanoi_981323>

1. Count consonants in a string (Iterative and recursive methods)

<https://ide.geeksforgeeks.org/36e1c555-bd6f-41e8-87a6-f5d7a0807dd9>

1. Recursive Insertion Sort
2. Print a pattern without using any loop
3. Binary to Gray code using recursion
4. Program for Chocolate and Wrapper Puzzle
5. Identify all Grand-Parent Nodes of each Node in a Map
6. Josephus Problem

[https://www.codingninjas.com/codestudio/libra ary/josephus-problem](https://www.codingninjas.com/codestudio/library/josephus-problem)

1. Factorial of a large number recursively

<https://www.codingninjas.com/codestudio/library/program-to-find-factorial-of-a-large-number-recursively>

1. Count all positive integers having n digits and the absolute difference between any two adjacent digits is K

<https://www.codingninjas.com/codestudio/library/count-all-positive-integers-having-n-digits-and-the-absolute-difference-between-any-two-adjacent-digits-is-k>

**STACK QS**

1. Reverse a Stack

<https://www.codingninjas.com/codestudio/problems/reverse-stack-using-recursion_631875?leftPanelTab=3>

1. Sort a stack

<https://www.codingninjas.com/codestudio/problems/sort-a-stack_537?leftPanelTab=3>

1. Find the Winner of the Circular Game(Josephus problem)

<https://leetcode.com/problems/find-the-winner-of-the-circular-game/>

**PALINDROME PARTITIONS**

1. Given a string, print all palindromic partitions

<https://www.codingninjas.com/codestudio/library/how-to-find-all-the-palindromic-partitions-of-a-string>

1. Combination of strings(same as palindrome partitioning)

**EASY QS**

1. Reverse a string using recursion

<https://ide.geeksforgeeks.org/6de75722-c42f-4808-99ff-27f9cabc513>

1. Check if a number is Palindrome or not

<https://classroom.codingninjas.com/app/classroom/me/10273/content/172236/offering/2136890/problem/36>

1. Program for the length of a string using recursion
2. First uppercase letter in a string (Iterative and Recursive)
3. Program to find the minimum (or maximum) element of an array
4. Product of 2 Numbers using Recursion

**LINKED LIST**

1. Recursive function to delete a kth node from the linked list
2. Reverse a Doubly linked list using recursion
3. Find the middle of the singly linked list Recursively
4. Insert a node in a singly linked list at a given position

<https://www.codingninjas.com/codestudio/library/insert-a-node-in-a-singly-linked-list-at-a-given-position-using-recursion>

1. Add two numbers represented by a Linked List

<https://www.codingninjas.com/codestudio/library/add-two-numbers-represented-by-a-linked-list>

**TREES**

1. Print all leaf nodes of a Binary Tree from left to right

(*done till page 3 of Babbar*)

<https://drive.google.com/drive/folders/1ofCyUF5sYzAc7Sa6IfYV1qBg6xziejko>