ALGORITHMS

RECURSIVE ALGORITHMS, HOW TO SOLVE A PROBLEM USING RECURSIVE ALGORITHMS, SAMPLE PROBLEMS AND THEIR RECURSIVE SOLUTION, IMPLIMENTATION IN ANY PROGRAMMING LANGUAGE OF YOUR CHOICE.

Group 5

NAMES OF PATICIPANT OF GROUP 5 MEMBERS

|  |
| --- |
| Ebacha Nelson Awah. Leader |
| Nformi vitorine nkeh |
| Nehlum Britney tamon |

|  |
| --- |
| Ndengue desire |



RECURSIVE ALGORITHMS.

Recursion is the method of solving a computational problem where the solution of a problem depends on solutions to smaller instances of the same problem.

Recursion solve such recursive problems by using functions that call themselves from within their own code. Recursion is one of the central ideas of computer science. Most programing languages support recursion by allowing a function to call itself from within it own code. Some functional programming on recursion to repeatly call a code.

IMPORTANCE OF RECURSION IN COMPUTER SCIENCE.

Recursion is important in computer science and programming because it allows for elegant and efficient solutions to certain problems. It is a powerful tool for solving problems that can be broken down into smaller, similar problems.

CHARACTERISTICS OF RECURSIVE ALGORITHM.

1. It must call it self recursively
2. It must have a base case
3. It must change it state and move toward the base

Examples of recursive algorithm.

MERGE SORT AND QUICK SORT ARE EXAMPLES OF RECURSIVE SORTING ALGORITHM.

ADVANTAGES AND DISADVANTAGES OF RECURSIVE ALGORITHM.

1. SOME PROBLEMS ARE INHERENTLY RECURSIVE, SUCH AS GRAPH AND TREE VERSAL.
2. FOR RECURSIVE CASES, YOU ONLY NEED TO DEFINE THE BASE CASE AND RECURSIVE CASE SO THE CODE IS EASIER AND SHORTER THAN ITERATIVE CODES.
3. A recursive program has greater space requirements than the iterative program.

HOW TO SOLVE A PROBLEM USING RECURSIVE ALGORITHMS.

To solve a problem using recursive algorithm, the following steps should be used as seen below.

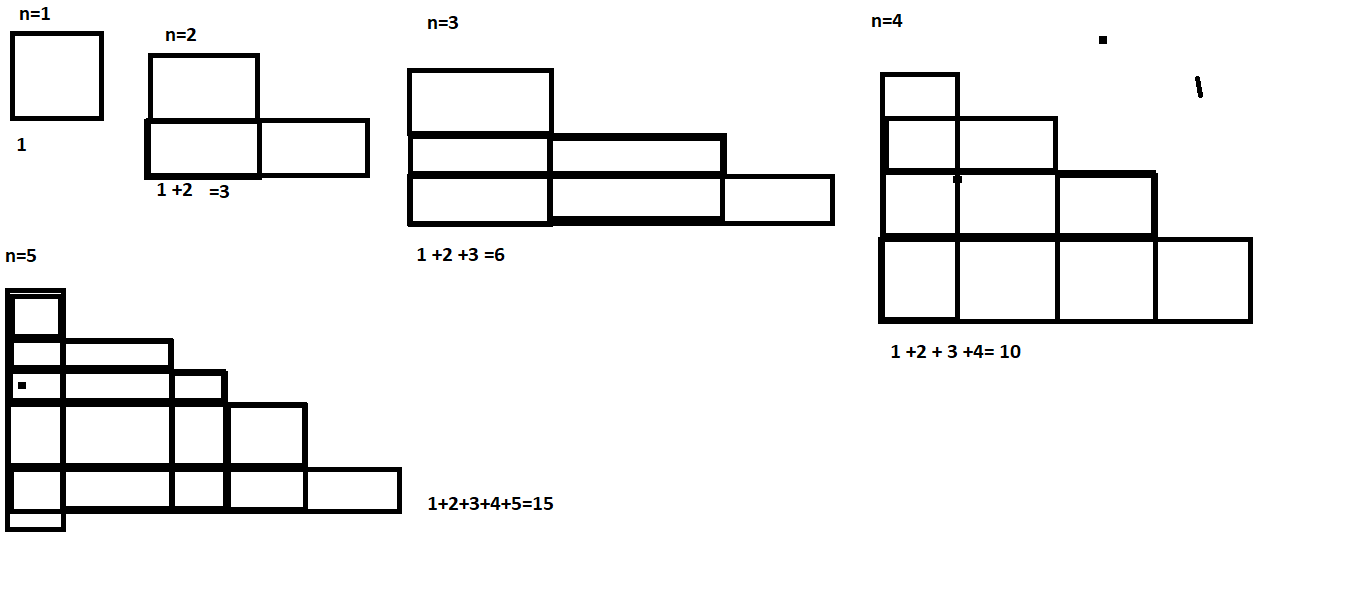
1. We first find what is the simplest possible input?
2. Play around with examples and visualized.
3. Relate had cases to simple cases
4. Generalise the pattern.
5. Write code by combining recursive patterns with base cases.

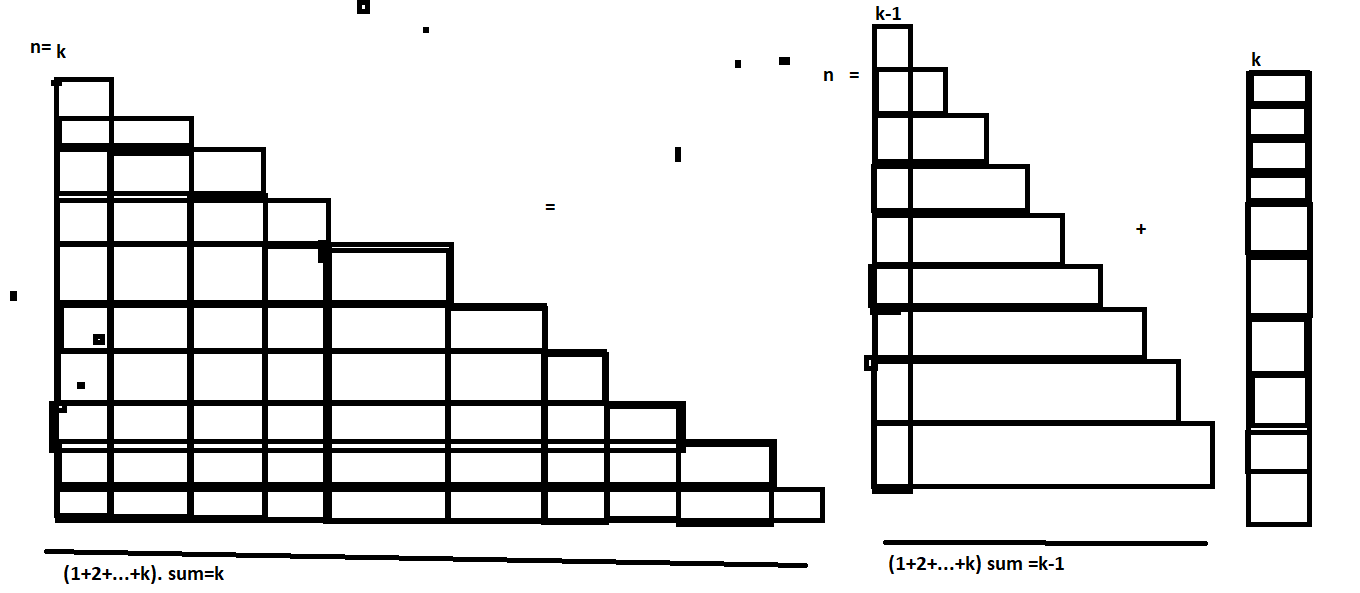
SAMPLE PROBLEMS AND THEIR RECURSIVE SOLUTIONS.

EXAMPLE. Write a recursive function that gives an input n sum all negative integers up to n.

Sum (0)->0

Base case.





Code. Sum(n) =sum (n-1) +n.

sum(n) = 0 if n=o

sum(n-1)+n

def sum(n):

if n =0:

return 0

else:

return n + sum(n -1)