Q. Write a program SumMain, that includes a recursive method computing the sum of the integers 1, 2, 3, ..., N. The computation should be based on the following principle: the sum of the integers from 1 to N is equal to the sum of the integers from 1 to N/2 added with the sum of the integers from N/2+1 to N.
Is this a good solution strategy? Motivate your answer!

Answer

f(N)=1+2+3+. . .+(N-1) +N

sum(N)=sum(N-1)+N

or

sum(N)=SUM(1,N/2)+SUM(N/2+1,N) where

SUM(1,N/2)= SUM(1,N/2-1)+N/2 and

SUM(N/2+1,N)= SUM(N/2+1,N-1)+N

For me the second strategy is good because

1. A single task is divided into two parts and delegates the task for others to finish it. And I would say this is the good strategy for efficiency concern as compare to doing the whole task once. Thinking about efficiency is a good strategy while designing or writing code. It is advisable to use iterative than recursive since performance is a big issue but in some cases recursive is the best candidate to solve problems easily as compared to iterative, good example for this is binary tree. For instance, in binary tree the root node delegate the task for left and/or right tree node. So what I think is dividing the task into parts and give it others to complete it in order to make fast. Computer usually divides the task into sub parts or different parts then delegates others to make the job done fast. Good example are, parallel programming, operating system, some algorithm like Merge sort etc. In parallel programming or operating system or merge sort tasks are subdivided into different parts and let others to process and finish it fast. Merge sort is efficient, as compare to other sorting algorithms that sorts the whole input without divides it into subparts, since it split the task into subparts until we get the indivisible part then sort each part then merge them to get the whole part.