A string (word) which reads the same from left to right and right to left is called a palindrome. Spaces, punctuation and upper/lower case does not matter. For example the following are palindromes: Anna, A lad named E. Mandala, A Santa dog lived as a devil God at NASA.

Write an algorithm which checks whether a string is a palindrome. Write your answer using pseudocode. If necessary, you can assume that you are given a separate algorithm/function which you can call and which converts the string into a suitable form (strips spaces, punctuation, all to lower case letters).

S = convert(string)

N = S.length

for i := 1 to floor(N/2) do

if S[i] != S[N-(i-1)] then

stop and print(“string is not a palindrome!”)

else

if i == floor(N/2) then

print(“string is a palindrome!”)

end

end

end

You have a bag full of gold coins. You know that one coin is fake, and that the fake coin is heavier than others (which all have the same weight). All coins look indentical, but you have scales that you can use to compare the weight of two sets of coins (which set is heavier or are they of equal weight). Write an algorithm in pseudocode to find the fake coin. Would recursion speed up the solution? How?

If the number of coins is even number, we divide coins into two groups and compare their weights. If It is an odd number, we exclude one of the coins, and divide the remaining coins into two equal group and weight them. If they have similar weight, the excluded coin is fake. Otherwise, we pick the heavier group and divide it into two groups and again repeat the above procedure and compare these subgroups weights. We continue dividing the groups into smaller subgroups, until the subgroups contain only 1 element. Recursion can speed up the solution by breaking the problem into similar but smaller problem and repeating it.

function Weighting(N)

{

Divide(N)

if (N<2) then

return

else

Pick coins from heavier side

Weighting (N/2)

}