1. **Write an algorithm for finding a palindrome. (Use recursive method).**

**Step1:** Take input as string.

**Step2:** To check palindrome reverse string using recursion and check both are equal

**Step3:** Take method named as palindrome and pass the input string as the parameter and then call reverse method uses recursion to reverse the string.

**Step4:** In the reverse method first check if the string is null , if it is null then it don't need to be reversed.

if input is null

return input

**Step5:** else extract the last character of the string and pass the remaining string using substring method using recursion and at last the string becomes empty and return

the reversed string.

else

return input.charAt(input.length()- 1) + reverse(input.substring(0, input.length() - 1))

**Step6:** In the palindrome method store this returned string in reverseinput and check both input string and returned string are equal.

**Step7:** If both strings are equal return "String is Palindrome".

if input and reverseinput are equal

then String is Palindrome

**Step8:** Else return "String is not Palindrome".

**2) Write an algorithm for generating anagrams for user entered input.**

For Generating Anagrams -- Permutation of a string using recursion

**Step1:** Take input as a string

**Step2:** Take one method named stringpermutation() and pass one empty string named permutation and input string and call recursively that method

**Step3:** In stringpermutation()(permutation, input) method call recursively stringpermutation () method for finding the Permutations of the string

if input.length is equal to zero

then print input string

else

for each i = 0 to input.length()

return stringpermutation (permutation + input.charAt(i),

input.substring(0 , i) + input.substring(i+1, input.length()));

**Step4 :** If input length becomes empty then it will print the permutation of the string.

**3) Write an algorithm for binary search.**

**Step1:** Take 5 numbers(array) as input and one input for searching .

**Step2:** For Binary search all numbers to be in sorting order.

**Step3:** For sorting I used bubble sort.

**Step4:** In bubble sort compare each element with other elements and do swap operation.

Bubble sort :

for each i = 0 to input.length-1

for each j = 0 to input.length-1

if input[j] > input[j+1]

swap input[j] and input[j+1]

**Step5:** After all the elements are sorted then find the middle element.

middle = lowerindex + (higherindex - lowerindex) /2

**Step6:** First check the given input is equal,greater or lesser with the middle element.

**Step7:** If the given number is equal then it is ok,or if the number is greater the middle element then it will check in rightside otherwise it will

check in leftside.

while input not found

if(array[middle] == input)

stop :Number found

else if array[middle] > input

higherindex = middle- 1

else if array[middle] < input

lowerindex = middle + 1

else element is not found

**Step8:** This process is repeated until the given number is found.

So by using Binary Search the time complexity is decreased since at every step half of the numbers are eliminated.