Recursion

Basics Problem on Recursion

Problem 1

Given an unsorted array of **N** elements and an element **X**. The task is to write a recursive function to check whether the element X is present in the given array or not.

Example:

```
array[] = {1, 2, 3, 4, 5}
X = 3.

The function should return True, as 3 is
present in the array.
```

Solution: The idea is to compare the first element of the array with X. If the element matches with X then return True otherwise recur for the remaining part of the array.

The **recursive function** will somewhat look like as shown below:

```
// arr[] is the given array
// l is the lower bound in the array
// r is the upper bound
// x is the element to be searched for
// l and r defines that search will be
// performed between indices l to r

bool recursiveSearch(int arr[], int l,
                            int r, int x)
{
    if (r < l)
        return false;
    if (arr[l] == x)
        return true;
    if (arr[r] == x)
        return true;

    return recursiveSearch(arr, l + 1,
                              r - 1, x);
}
```

Time Complexity: The above algorithm runs in O(N) time where, N is the number of elements present in the array.
Space Complexity: There is no extra space used however the internal stack takes O(N) extra space for recursive calls.

Problem 2

Given a string, the task is to write a recursive function to check if the given string is palindrome or not.

Examples:

```
Input : string = "malayalam"
Output : Yes
Reverse of malayalam is also
malayalam.

Input : string = "max"
Output : No
Reverse of max is not max.
```

Solution: The idea to write the recursive function is simple and similar to the above problem:

1. If there is only one character in string, return true.
2. Else compare first and last characters and recur for remaining substring.

Recursive Function:

```
// s and e defines the start and end index of string

bool isPalindrome(char str[], int s, int e)
{
    // If there is only one character
    if (s == e)
        return true;

    // If first and last
    // characters do not match
    if (str[s] != str[e])
        return false;

    // If there are more than
    // two characters, check if
    // middle substring is also
    // palindrome or not
    if (s < e)
        return isPalindrome(str, s + 1, e - 1);

    return true;
}
```