Day 80/180 Recursion in Binary Search

1: Given an array in non-increasing order, an element is given X, find if that element is present in the array or not. print 1 if it is present else print 0.

Logic and code are discussed in the video.

2: Write a recursive function to reverse the elements of an array.

- As we know to reverse an array we just need to swap the elements of the array that are equidistant from the mid.
- So, we're swapping the last ones in recursion and sending the remaining to the recursion to reverse.

```
#include <iostream>
using namespace std;

// Function to reverse elements in the array
void reverse(int arr[], int i, int j)
{
    // Base case: If the start index is greater than or equal to
the end index, return
    if (i >= j)
        return;

    // Swap the elements at the start and end indices
    int temp = arr[i];
```

```
    arr[i] = arr[j];
    arr[j] = temp;

    // Recursively call the reverse function for the remaining
elements
    reverse(arr, i + 1, j - 1);
}

int main() {
    // Array to be reversed
    int arr[] = {1, 2, 3, 4, 5};

    // Number of elements in the array
    int n = 5;

    // Call the reverse function to reverse the array
    reverse(arr, 0, n - 1);

    // Display the reversed array
    cout << "Reversed Array: ";
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }

    return 0;
}
```

3: Write a recursive function to rotate elements in an array to the right by 1 position.

- We have to rotate the array to the right by 1 position, that is shifting (i)th position elements to (i+1)th position.

- We’re making a recursive call to place the previous element in the current index.
- Sending to the current index value to recursion for the remaining.
- For the last case, we know that it will be positioned at the first place “0th” index of the array.

```
#include <iostream>
using namespace std;

// Function to rotate elements in an array to the right by one position
void rotateRightByOne(int arr[], int index, int val, int n) {

    // Base case: If the entire array has been processed
    if (index == n) {
        // Place the last element at the beginning
        arr[0] = val;
        return;
    }

    // Save the current element at the current index
    int temp = arr[index];

    // Update the current element with the provided value
    arr[index] = val;

    // Recursively call the function for the next index with the original
value
    rotateRightByOne(arr, index + 1, temp, n);
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = 5;

    // Display the original array
    cout << "Original Array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
```

```
    }

    // Rotate the array to the right by 1 position
    rotateRightByOne(arr, 0, -1, n);

    // Display the rotated array
    cout << "\nRotated Array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}
```