Binary Search:

Binary Search is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(log N).

Conditions:

To apply Binary Search algorithm:

- The data structure must be sorted.
- Access to any element of the data structure takes constant time.

Algorithm:

1. Divide the search space into two halves by finding the middle index “mid”
2. Compare the middle element of the search space with the key.
3. If the key is found at middle element, the process is terminated.
4. If the key is not found at middle element, choose which half will be used as the next search space.
5. If the key is smaller than the middle element, then the left side is used for next search.
6. If the key is larger than the middle element, then the right side is used for next search.
7. This process is continued until the key is found or the total search space is exhausted.

Code:

```
// C++ program to implement iterative Binary Search
#include <bits/stdc++.h>
using namespace std;

// An iterative binary search function.
int binarySearch(int arr[], int l, int r, int x)
{
        while (l <= r) {
                int m = l + (r - l) / 2;

                // Check if x is present at mid
```

```
            if (arr[m] == x)
                return m;

            // If x greater, ignore left half
            if (arr[m] < x)
                l = m + 1;

            // If x is smaller, ignore right half
            else
                r = m - 1;
        }

        // If we reach here, then element was not present
        return -1;
}

// Driver code
int main(void)
{
        int arr[] = { 2, 3, 4, 10, 40 };
        int x = 3;
        int n = sizeof(arr) / sizeof(arr[0]);
        int result = binarySearch(arr, 0, n - 1, x);
        (result == -1)
            ? cout << "Element is not present in array"
            : cout << "Element is present at index " << result;
        return 0;
}
```

Output:

```
PS D:\Mirror\ICT\3rd YEAR\SEM 5\Desig
-vscode.cpptools-1.17.5-win32-x64\debu
'--stdout=Microsoft-MIEngine-Out-ysh3
id-gmukj311.5ij' '--dbgExe=D:\Mirror\
Element is present at index 1
PS D:\Mirror\ICT\3rd YEAR\SEM 5\Desig
```

Space complexity: _____

Justification:_____
_____

Time complexity:

Best case time complexity: _____

Justification:_____
_____

Worst case time complexity: _____

Justification:_____