ALGORITHMS AND DATA STRUCTURES

CS106.3

1. **What is Binary Search.**

    - Binary Search is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half.

2. **What are the advantages and disadvantages of Binary Search.**

    Advantages:

    - Binary search is one of the quickest looking through calculations.
    - It is utilized for tracking down the area of a component in a direct cluster.
    - It chips away at the guideline of gap and overcomes strategy.
    - It's a genuinely straightforward calculation, whenever comprehended properly
    - It's notable and frequently executed for you as a library schedule
    - It is preferred for large size data sets
    - It is more efficient for large size data
    - It is implemented on a multidimensional array

    Disadvantages:

    - It's more confounded than straight inquiry and is over the top excess for tiny quantities of components.
    - It works just on records that are arranged and kept arranged. That isn't generally feasible, particularly assuming components are continually being added to the rundown.
    - It works just on component types for which there exists not as much relationship. A few kinds basically can't be arranged

3. **Write the Algorithm of Binary Search.**
    - Divide the search space into two halves by finding the middle index "mid".
    - Compare the middle element of the search space with the key.
    - If the key is found at middle element, the process is terminated.
    - If the key is not found at middle element, choose which half will be used as the next search space.
    - If the key is smaller than the middle element, then the left side is used for next search.
    - If the key is larger than the middle element, then the right side is used for next search.
    - This process is continued until the key is found or the total search space is exhausted.

4. **Write a simple C program for Binary Search**

```
// 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 n = sizeof(arr) / sizeof(arr[0]);
    int x = 10;
    int result = binarySearch(arr, 0, n - 1, x);
    (result == -1) ? printf("Element is not present"
                            " in array")
                   : printf("Element is present at "
                            "index %d",
                            result);
    return 0;
}
```