Assignment No: 1

Name: Kedar Kale

Batch: B12

Roll No: SCETTYB129

Problem Statement: Design and implement Binary search algorithm using Divide and Conquer method for a given input. Determine the time required to search an element. also implement randomized algorithm of binary search.

Algorithm:

```
Procedure binary_search
   A ← sorted array
   n ← size of array
   x ← value to be searched

   Set lowerBound = 1
   Set upperBound = n

   while x not found
      if upperBound < lowerBound
         EXIT: x does not exists.

      set midPoint = lowerBound + ( upperBound - lowerBound )
/ 2

      if A[midPoint] < x
         set lowerBound = midPoint + 1

      if A[midPoint] > x
         set upperBound = midPoint - 1

      if A[midPoint] = x
         EXIT: x found at location midPoint
   end while

end procedure
```

Program Execution:

1) **Without using Randomized Algorithm**

Program:

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

int BinarySearch(int list[], int low, int high, int key);


int main()
{
    int SIZE;
    int Array[SIZE];
    int value;
    int i,j,temp;
    cout << "Enter the size of your array. \n";
    cin >> SIZE;
    cout<< "\n";

    cout << "Enter the numbers of your array. \n";
    for(int i= 0; i< SIZE; i++)
        {
            cout << "Element " <<i+1 <<": ";
            cin>> Array[i];
        }

    for(i=1;i<SIZE;i++)
        {
                for(j=0;j<(SIZE-i);++j)
                        if(Array[j]>Array[j+1])
                        {
                                temp=Array[j];
                                Array[j]=Array[j+1];
                                Array[j+1]=temp;
                        }
        }

        cout<<"\nArray after sorting:";
        for(i=0;i<SIZE;i++){
                cout<<" "<<Array[i];
        }


    cout << "\nEnter a number to find from the above list of numbers. \n";
    cin >> value;
    cout << "\n\n";

    int Loc = BinarySearch(Array, 0, SIZE- 1, value);

    if(Loc >= 0)
        {
        cout<< "Number is found.\n";
        cout << "The number is at location: " <<Loc +1 << ".\n";
        }
        else
            cout<< "Number is not found.\n";
```

```
    return 0;
}

int BinarySearch(int list[], int low, int high, int key)
{
    if(low > high)
    {
        return -1;
    }
    int mid= low + (high - low)/2;

        if (list[mid] == key)
        {
            return mid;
        }
        else if (list[mid] <key)
            return BinarySearch(list,low,mid -1, key);
        else
            return BinarySearch(list,mid + 1, high, key);
}
```

Input/Output:

```
Enter the size of your array. 5

Enter the numbers of your array.

Element 1:10
Element 2:45
Element 3:66
Element 4:55
Element 5:4

Array after sorting: 4 10 45 55 66

Enter a number to find from the above list of numbers. 10

Number is found.

The number is at location: 2

Enter a number to find from the above list of numbers. 20

Number is not found.
```

2) Using Randomized Algorithm

Program:

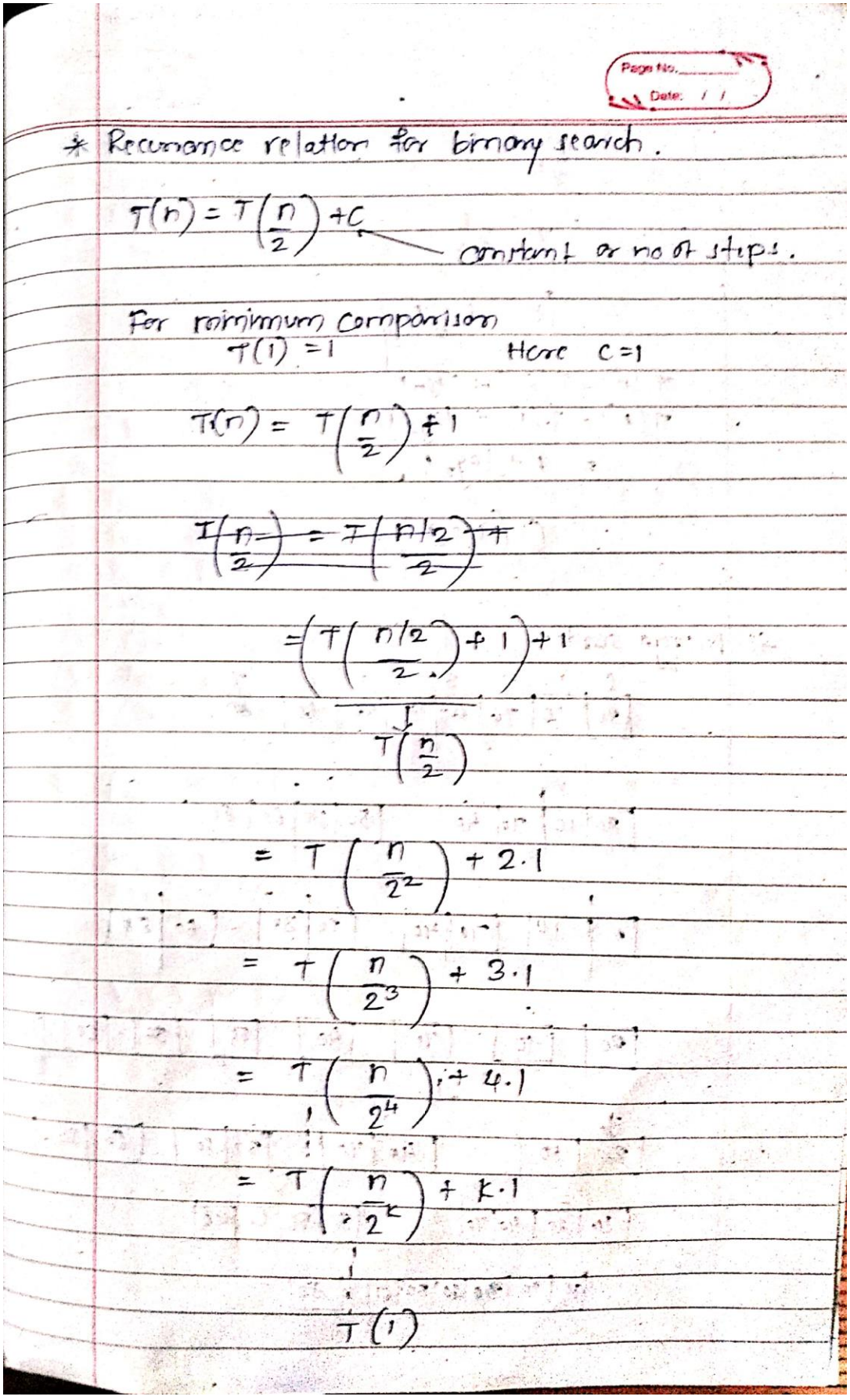
```
#include <iostream>
#include <ctime>
using namespace std;

// To generate random number
// between x and y ie.. [x, y]
int getRandom(int x, int y)
{
    srand(time(NULL));
    return (x + rand() % (y-x+1));
}

// A recursive randomized binary search function.
// It returns location of x in
// given array arr[l..r] is present, otherwise -1
int randomizedBinarySearch(int arr[], int l,
                           int r, int x)
{
    if (r >= l)
    {
        // Here we have defined middle as
        // random index between l and r ie.. [l, r]
        int mid = getRandom(l, r);

        // If the element is present at the
        // middle itself
        if (arr[mid] == x)
            return mid;

        // If element is smaller than mid, then
        // it can only be present in left subarray
        if (arr[mid] > x)
          return randomizedBinarySearch(arr, l,
                                mid-1, x);

        // Else the element can only be present
        // in right subarray
        return randomizedBinarySearch(arr, mid+1,
                                      r, x);
    }

    // We reach here when element is not present
    // in array
    return -1;
}

// Driver code
int main()
{
    int arr[50];
    int n,i;
    int x,v;
    cout<<"\nEnter the size of the array ";
    cin>>n;
    cout<<"\nEnter the elements into the array \n";
    for(i=0;i<n;i++){
```

```
        cout<<"Element "<<i+1<<" ";
        cin>>arr[i];
    }
    cout<<"\nEnter the element that you want to search ";
    cin>>x;
    v = x;
    int result = randomizedBinarySearch(arr, 0, n, v);
    (result == -1)? cout<<"Element is not present in array"
    : cout<<"Element is present at index "<<result + 1;
    return 0;
}
```

Input/Output:

```
Enter the size of the array 5



Enter the elements into the array

Element 1 33
Element 2 22
Element 3 3
Element 4 2
Element 5 678

Enter the element that you want to search 3

Element is present at index 3
```

Time complexity:



* Recurrence relation for binary search.

$T(n) = T\left(\frac{n}{2}\right) + C$ — constant or no of steps.

For minimum comparison

$T(1) = 1$ Here $C = 1$

$T(n) = T\left(\frac{n}{2}\right) + 1$

~~$T\left(\frac{n}{2}\right) = T\left(\frac{n/2}{2}\right) +$~~

$= \left(T\left(\frac{n/2}{2}\right) + 1\right) + 1$

$T\left(\frac{n}{2}\right)$

$= T\left(\frac{n}{2^2}\right) + 2 \cdot 1$

$= T\left(\frac{n}{2^3}\right) + 3 \cdot 1$

$= T\left(\frac{n}{2^4}\right) + 4 \cdot 1$

$= T\left(\frac{n}{2^k}\right) + k \cdot 1$

$T(1)$

Page No.
Date: / /

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$\boxed{k = \log_2 n.}$$

$$T(n) = T\left(\frac{n}{2^k}\right) + \log_2 n.$$

$$T(n) = T(1) + \log_2 n$$

$$= 1 + \log_2 n$$

$$O(\log_2 n).$$