Q2 Implement Linear Search and Binary Search. Analyze their time complexities.

TIME COMPLEXITY:
1. Binary Search: O(log(n)) (valid for only ordered lists)
1. Linear Search: O(n)

SPACE COMPLEXITY: O(1)

```
#include <iostream>
#include <vector>
#include <random>
#include <chrono>

using namespace std;


int linearSearch(vector<int> & A, int val)
{
    for(int i = 0; i < A.size(); ++i)
        if(A[i] == val)
            return i;
    return -1;
}

int binarySearch(vector<int> & A, int start, int end, int val)
{
    if(start > end)
        return -1;
    int mid = start + (end - start) / 2;
    if(val < A[mid])
        return binarySearch(A, start, mid - 1, val);
    else if(val > A[mid])
        return binarySearch(A, mid + 1, end, val);
    else return mid;
}

int main()
{
    vector<int> A(1000);
    for(int i = 0; i < 1000; ++i)
        A[i] = rand() % 10000;
    sort(A.begin(), A.end());
    cout << "INPUT ARRAY:\n";
    for(int i = 0; i < 1000; ++i)
        cout << A[i] << " ";
    A[998] = 9940;
    cout << endl << endl;
    auto start_linear = chrono::high_resolution_clock::now();
    cout << "Linear Search index for 9940: " << linearSearch(A, 9940) <<
     endl;
    auto end_linear = chrono::high_resolution_clock::now();
    cout << "Binary Search index for 9940: " << binarySearch(A, 0,
     int(A.size()) - 1, 9940) << endl << endl;
```

```
    auto stop = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::microseconds>(end_linear
     - start_linear);
    cout << "TIME TAKEN(LINEAR SEARCH): " << duration.count() << "
     microsecond" << endl;
    duration = chrono::duration_cast<chrono::microseconds>(stop -
     end_linear);
    cout << "TIME TAKEN(BINARY SEARCH): " << duration.count() << "
     microsecond" << endl;
}
```

OUTPUT:

```
INPUT ARRAY:
0 33 98 2217 7533 7544 7561 7584 7633 7638 7656 7670 7683 7692 7696 7698
 7699 7704 7709 7709 7719 7722 7730 7732 7746 7747 7758 7773 7781 7787 7801
 7826 7827 7844 7844 7844 7854 7865 7874 7875 7876 7878 7879 7904 7920 7923
 7929 7936 7939 7942 7944 7947 7968 7972 7987 7992 7995 8008 8013 8014 8015
 8024 8028 8034 8048 8060 8080 8111 8124 8138 8142 8144 8150 8155 8163 8165
 8177 8192 8209 8211 ... 9358 9364 9378 9397 9415 9419 9425 9451 9453 9461
 9498 9503 9505 9506 9517 9518 9530 9531 9536 9543 9549 9551 9553 9560 9565
 9579 9589 9590 9593 9594 9603 9605 9627 9649 9651 9662 9669 9684 9715 9719
 9745 9759 9789 9790 9798 9814 9816 9818 9821 9824 9847 9852 9860 9870 9874
 9879 9879 9880 9891 9901 9911 9915 9940 9943 9949 9955 9956 9970 9997

Linear Search index for 9940: 993
Binary Search index for 9940: 993

TIME TAKEN(LINEAR SEARCH): 9 microsecond
TIME TAKEN(BINARY SEARCH): 4 microsecond
Program ended with exit code: 0
```