Assignment III

Name=Abhay kumar

Roll no. =2

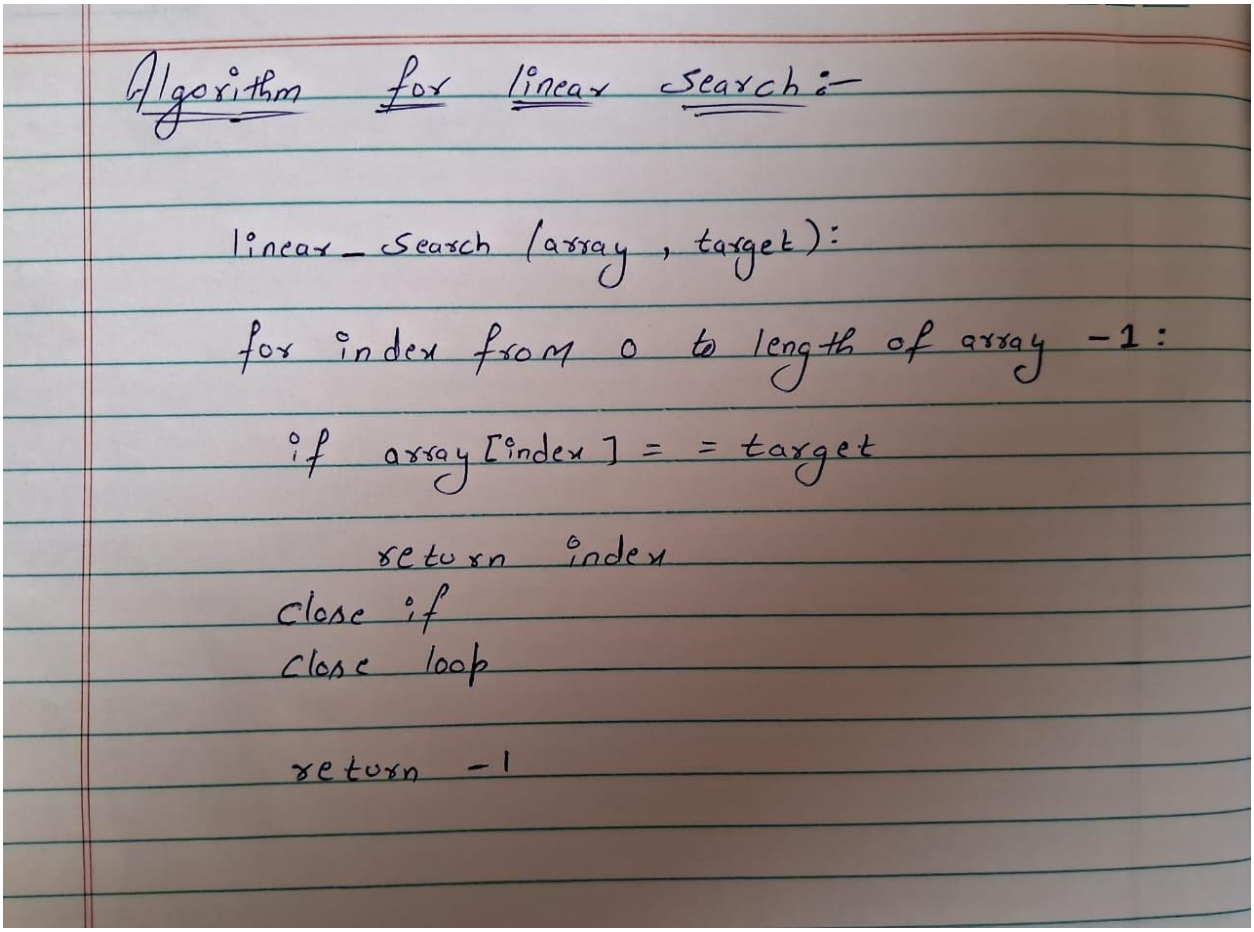
Branch=computer engineering

Subject= DAA

Aim: Write an algorithm for Linear Search and Binary Search. Write a program to solve given problem using your algorithms. Apply coding style in your programs.

Linear search:

Algorithm:



Test cases:

Date:

Test Cases for the linear Search:

Positive Test Cases :-

① Input : Element [ ] = { 2 , 4 , 6 , 8 , 10 }
Key = 6
output : The Key is present at index : 2
Expected out : The Key is present at index : 2

② Input : Element [ ] = { 1 , 3 , 5 , 7 , 9 }
Key = 7
output : The Key is present at index : 3
Expected output : The Key is present at index : 3

③ Input : Element [ ] = { 10 , 20 , 30 , 60 , 40 }
Key = 40
output : The Key is present at index : 4
Expected output : The Key is present at index : 4

Negative Test Cases :

① Input : Element [ ] = { 2 , 4 , 6 , 8 , 10 }
Key = 3
output : The Key is not present.
Expected : The Key is not present

(2) Input : Element [] = { 1 , 3, 5, 7, 9 }
Key = 2

output = the key is not present.

Expected = The key is not present.

(3) Input : Element [] = { 5 2 3 7 1 }
Key = 4

output : The key is not present.

Expected : The key is not present.

Program:

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

int search(int arr[],int size,int key)
{
    for(int i=0; i<=size; i++)
    {
        if (arr[i]==key)
        {
            return key;
        }
    }
    return -1;
}

int main()
{
    int arr[5]={2,4,6,8,10};
    cout<<"Enter the element to search for"<<endl;
    int key;
    cin>>key;
    int found=search(arr,10,key);
    if (found){
        cout<<"The key is present"<<linearfound<<endl;

    }
    else{
        cout<<"The key is not present"<<endl;
    }
    return 0;
}
```

Output:

```
arr[]={10,20,30,40,60}
Enter the element to search for
40
The key is present at index:    3
```

```
arr[]={2,4,6,8,10}
Enter the element to search for
6
The key is present at index:    2
```

```
C:\Users\abhay choudhary\OneDrive\Documents\.vscode frb\Untitled1.exe
arr[]={1,3,5,7,9}
Enter the element to search for
2
The key is not present
```

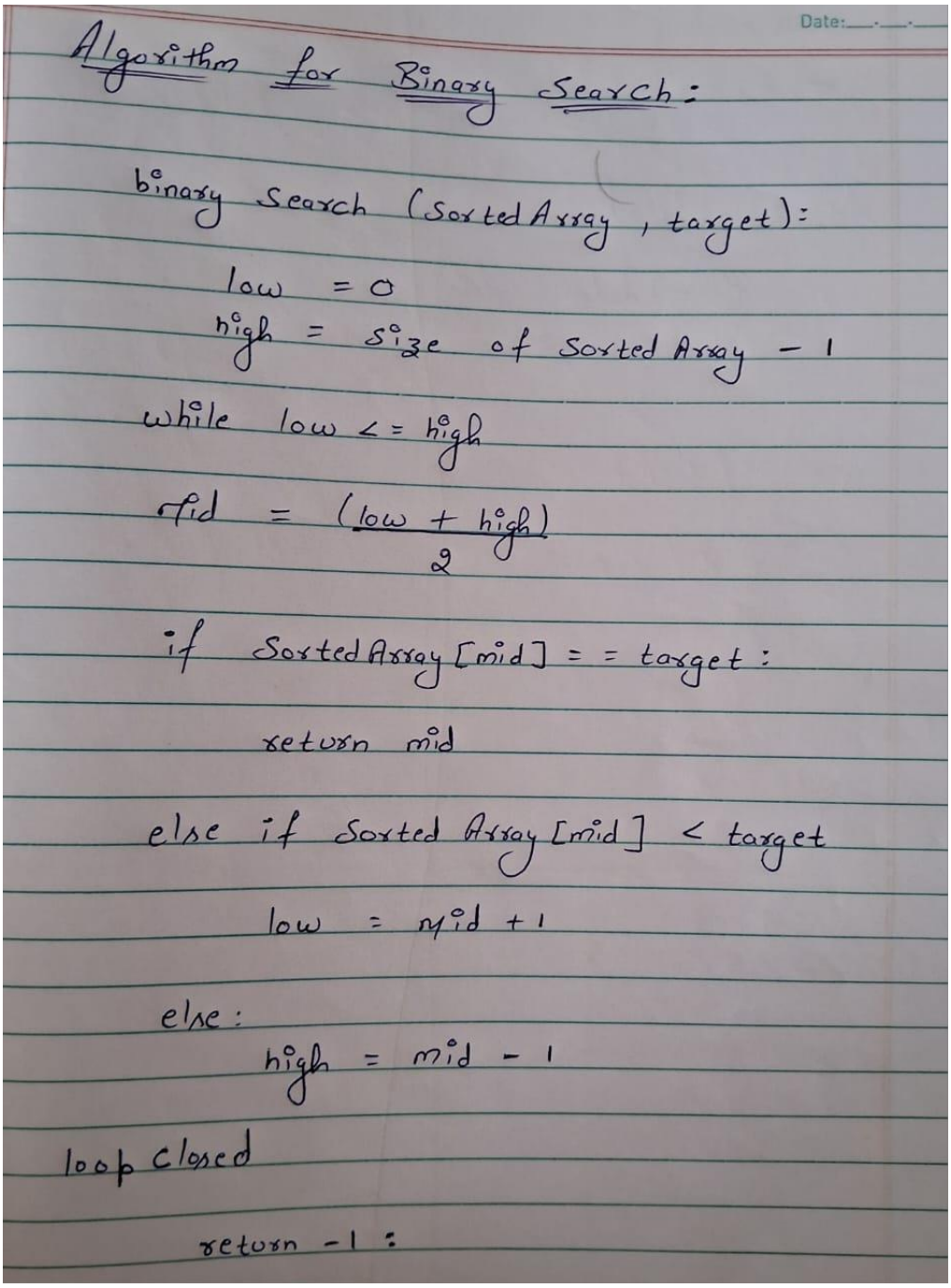
```
arr[]={2,4,6,8,10}
Enter the element to search for
3
The key is not present
```

```
arr[]={1,3,5,7,9}
Enter the element to search for
7
The key is present at index:    3


--------------------------------
Process exited after 16.41 seconds with return value 0
Press any key to continue . . .
```

Binary Search:

Algorithm:


Date:
Algorithm for Binary Search:
binary Search (Sorted Array, target):
low = 0
high = size of Sorted Array - 1
while low <= high
mid = (low + high) / 2
if Sorted Array[mid] == target:
return mid
else if Sorted Array[mid] < target
low = mid + 1
else:
high = mid - 1
loop closed
return -1 :

Test Cases:

Test cases for Binary Search:

Positive Test cases:

(i) Input: Element [] = {2, 4, 6, 8, 10}
Key = 6
output = The key is present at index: 2

Expected output :- The key is present at index: 2

(ii) Input: Element [] = {1, 3, 5, 7, 9}
Key = 9
output: The key is present at index: 4

Expected output :- The key is present at index: 4

(iii) Input: Element [] = {10, 20, 40, 60}
Key = 10
output: The key is present at index: 0

Expected output :- The key is present at index: 0

Negative Test cases :-

(i) Input: Element [] = {2, 4, 6, 8, 10}
Key = 3
output :- The key is not present.

Expected output :- The key is not present.

(ii) Input : Element [ ] = { 2, 3, 5, 7, 9 }

Key = 2

output : The key is not present.

Expected output : The key is not present.

(iii) Input : Element [ ] = { }

Key = 1

output : The key is not present.

Expected output : The key is not Present.

Program:

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

int Binarysearch(int arr[], int key, int size) {
    int start = 0, end = size - 1;

    while (start <= end) {
        int mid = start + (end - start) / 2;

        if (arr[mid] == key) {
            return mid;
        }
        else if (arr[mid] < key) {
            start = mid + 1;
        }
        else {
            end = mid - 1;
        }
    }
    return -1;
}
int main() {
    int arr[] = {1,2,3,4,5,6,7,8,9};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key;
    cout << " Enter the key to find: ";
    cin >> key;
    int output = Binarysearch(arr, key, size);
    if (output != -1) {
        cout << "The key is present at index: " << output << endl;
    }
    else {
        cout << "The key is not present" << endl;
    }
    return 0;
}
```

Output:

```
arr[]={10,20,30,40,60}
Enter the element to search for
40
The key is present at index:    3
```

```
arr[]={2,4,6,8,10}
Enter the element to search for
6
The key is present at index:    2
```

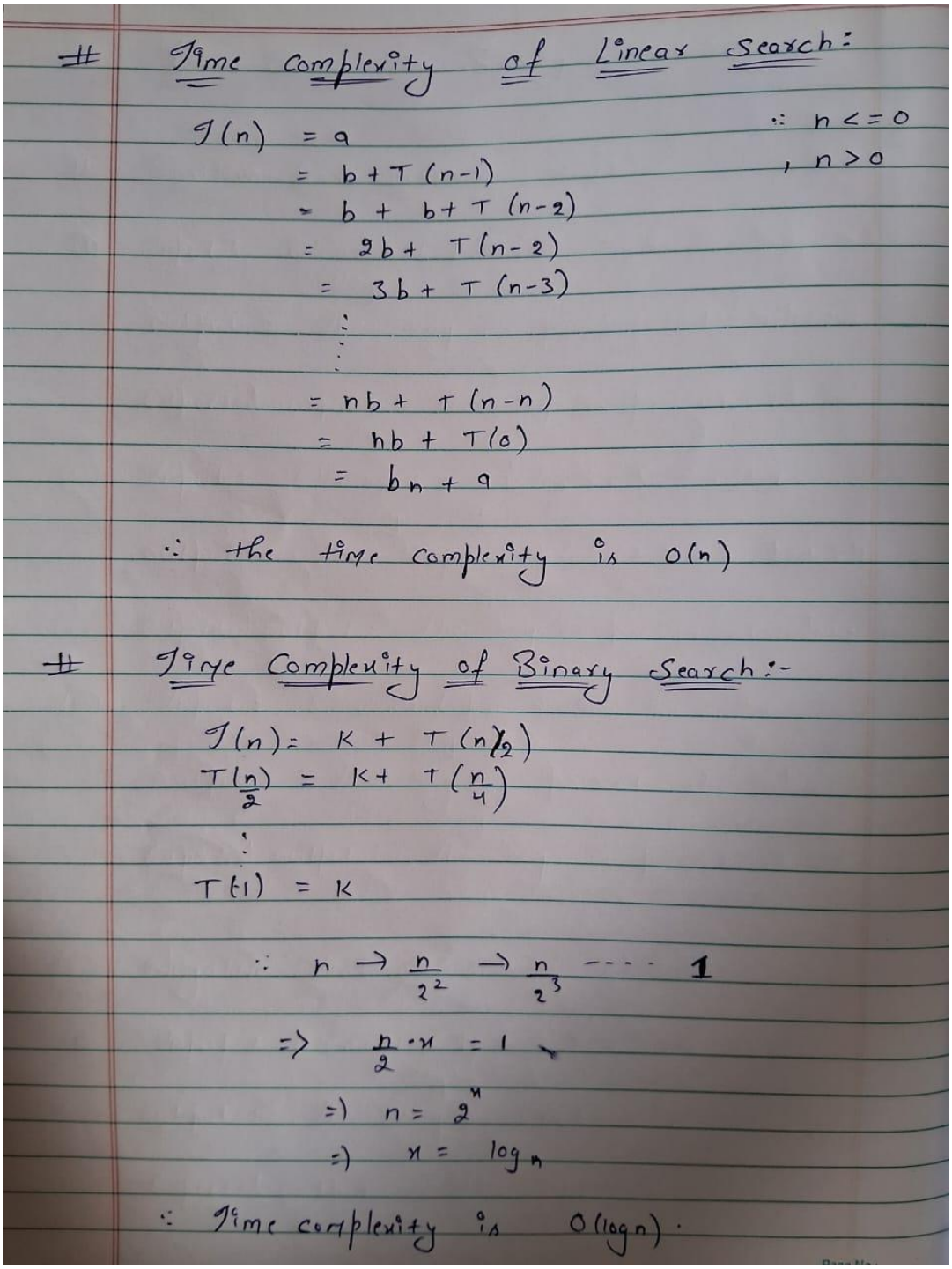
```
C:\Users\abhay choudhary\OneDrive\Documents\.vscode frb\Untitled1.exe
arr[]={1,3,5,7,9}
Enter the element to search for
2
The key is not present
```

```
arr[]={2,4,6,8,10}
Enter the element to search for
3
The key is not present
```

```
arr[]={1,3,5,7,9}
Enter the element to search for
7
The key is present at index:    3


--------------------------------
Process exited after 16.41 seconds with return value 0
Press any key to continue . . .
```

Time Complexity:



Time complexity of Linear Search:

$T(n) = a$ $\quad \because n <= 0$

$= b + T(n-1)$ $\quad , n > 0$

$= b + b + T(n-2)$

$= 2b + T(n-2)$

$= 3b + T(n-3)$

$\vdots$

$= nb + T(n-n)$

$= nb + T(0)$

$= bn + a$

$\therefore$ the time complexity is $O(n)$

Time Complexity of Binary Search:-

$T(n) = K + T(n/2)$

$T\left(\frac{n}{2}\right) = K + T\left(\frac{n}{4}\right)$

$\vdots$

$T(1) = K$

$\because n \rightarrow \frac{n}{2^2} \rightarrow \frac{n}{2^3} \quad \cdots \quad 1$

$\Rightarrow \frac{n}{2^x} = 1$

$\Rightarrow n = 2^x$

$\Rightarrow x = \log n$

$\therefore$ Time complexity is $O(\log n)$.

Conclusion:

- **Efficiency**: Binary search is significantly more efficient than linear search for large datasets due to its logarithmic time complexity, but it requires that the list be sorted.
- **Flexibility**: Linear search is more versatile since it doesn't require any specific ordering of the data, making it useful in a wider range of scenarios.

Choosing between the two depends largely on whether the data is sorted and the size of the dataset. For unsorted or small lists, linear search is simpler and sufficient. For large sorted lists, binary search offers much better performance.