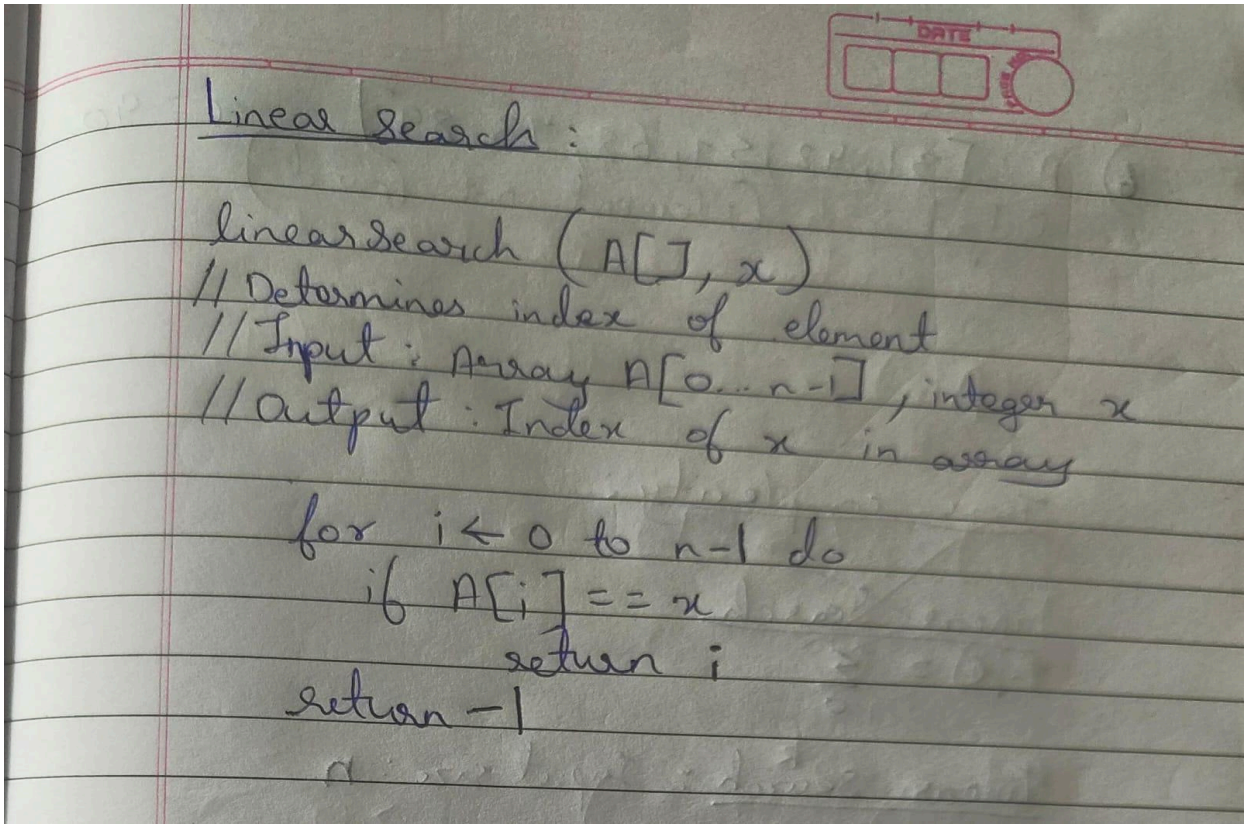
DAA Assignment 2

Ishaan Shaikh
231070063

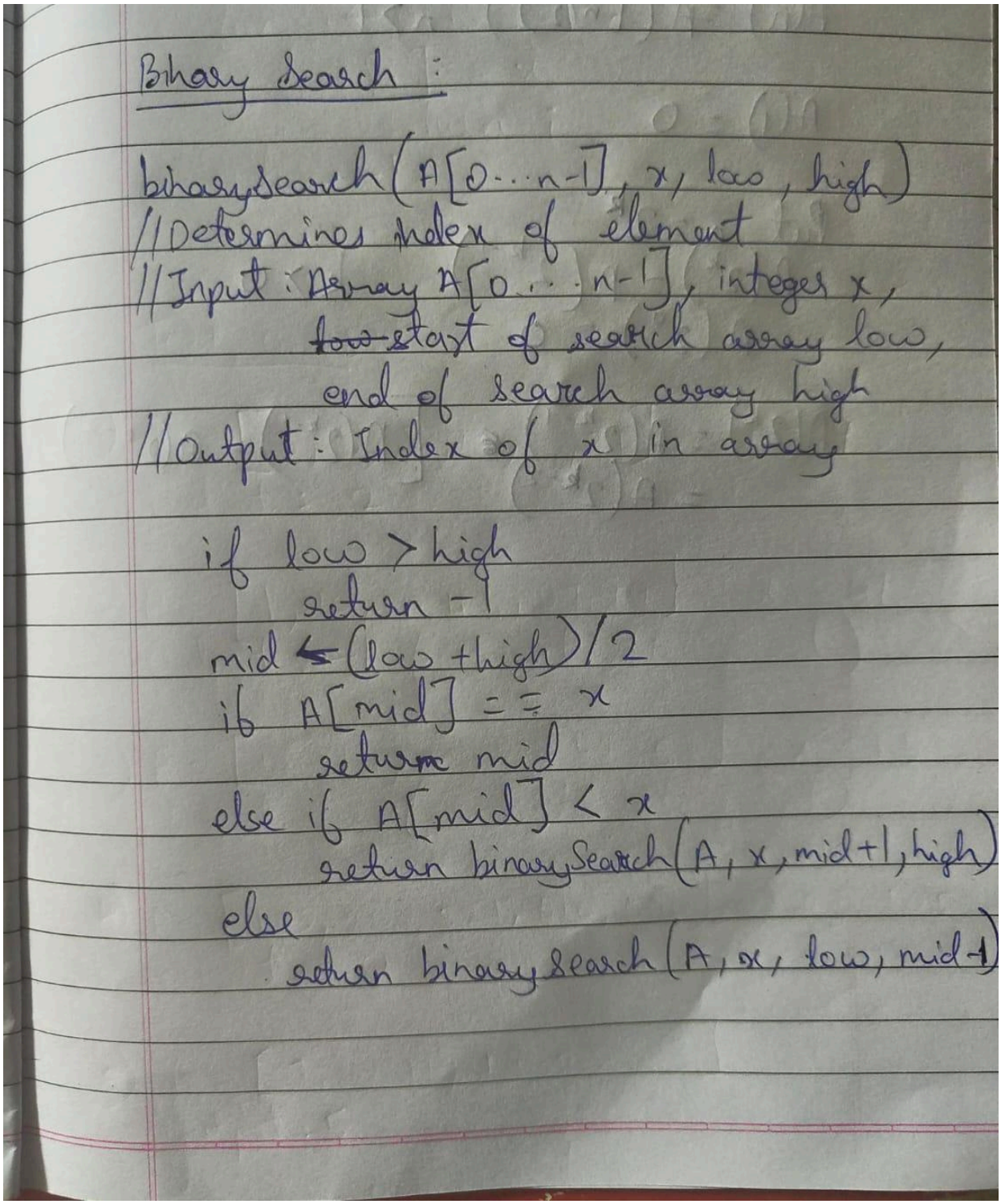
Q. Write an algorithm for Linear Search and Binary Search. Write a program to solve the given problem using your algorithms. Apply coding style in your programs.

Algorithm:

Linear Search:



Binary search:


Binary Search :
binarySearch(A[0...n-1], x, low, high)
//Determines index of element
//Input: Array A[0...n-1], integer x,
start of search array low,
end of search array high
//Output: Index of x in array
if low > high
return -1
mid ← (low + high)/2
if A[mid] == x
return mid
else if A[mid] < x
return binarySearch(A, x, mid+1, high)
else
return binarySearch(A, x, low, mid-1)

Test cases:

Test cases:

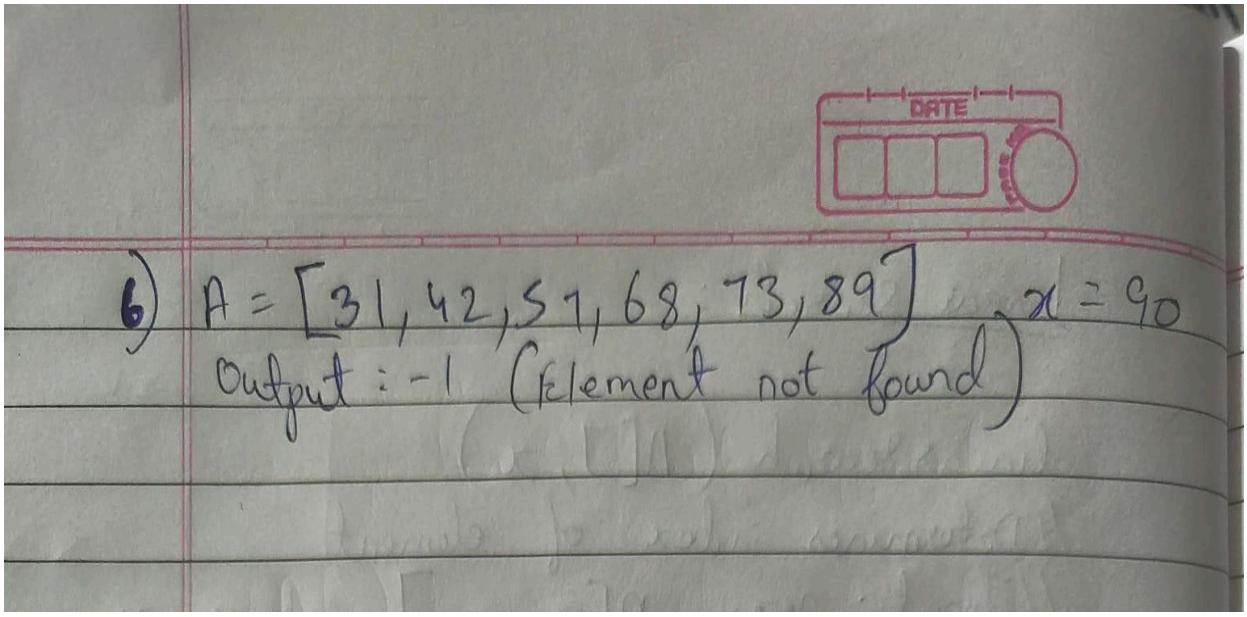
1) A = [3, 5, 9, 10, 15, 21] x = 9
Output : 2

2) A = [1, 10, 20, 42, 57, 99] x = 41
Output : -1 (Element not found)

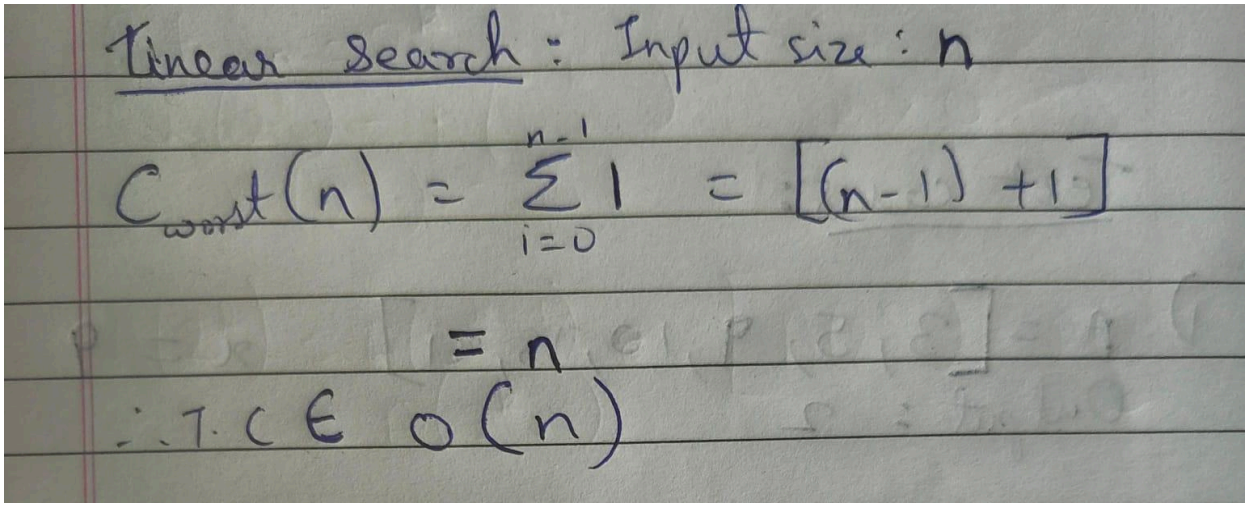
3) A = [12, 14, 20, 37, 41, 58] x = 15
Output : -1 (element not found)

4) A = [105, 210, 321, 410, 500] x = 105
Output : 0

5) A = [7, 10, 11, 19, 25, 33] x = 33
Output : 5



Time Complexity:



Binary Search:

Input size : n

Basic operation: Check conditions and make a recursive call

Let A(n) be time complexity of algorithm

$A(n) = A(n/2) + c$, $A(1) = c$

where c is the time required to check the conditions.

$A(n/2) = A(n/4) + c$

$\therefore A(n) = A(n/4) + 2c$

$A(n) = A(n/8) + 3c$ $[A(n/4) = A(n/8) + c]$

~~$A(n) = A(1) + kc$, k = constant~~

$A(n) = A(n/2^k) + kc$

$A(n) = A(1) + C$, C = constant

$A(n/2^k) = A(1)$

$n/2^k = 1$

$\therefore n = 2^k$

$\therefore \log_2 n = k$

$A(n) = A(n/2^{\log_2 n}) + c\log_2 n$

$= A(n/n) + c\log_2 n$

$= c + c\log_2 n$

$A(n) = c(1 + \log_2 n)$

$\therefore$ Time complexity $\in O(\log_2 n)$

Program:

```
def linear_search(arr, x):
        for i in range(len(arr)):
        if arr[i] == x:
        return i
        return -1

def binary_search(arr, x, low, high):
        if low > high:
        return -1
        mid = (high + low) // 2
        if arr[mid] == x:
        return mid
        elif arr[mid] > x:
        return binary_search(arr, x, low, mid - 1)
        else:
        return binary_search(arr, x, mid + 1, high)

def main():
        arr = []
        n = int(input("Enter number of elements: "))
        for i in range(n):
        arr.append(int(input(f"Enter element {i + 1}: ")))
        x = int(input("Enter element to search: "))
        print(f"Index of {x} using linear search: {linear_search(arr, x)}")
        print(f"Index of {x} using binary search: {binary_search(arr, x, 0, len(arr) -
1)}")

if __name__ == "__main__":
        main()
```

Output:

1.

```
Enter number of elements: 6
Enter element 1: 3
Enter element 2: 5
Enter element 3: 9
Enter element 4: 10
Enter element 5: 15
Enter element 6: 21
Enter element to search: 9
Index of 9 using linear search: 2
Index of 9 using binary search: 2
```

2.

```
Enter number of elements: 6
Enter element 1: 1
Enter element 2: 10
Enter element 3: 20
Enter element 4: 42
Enter element 5: 57
Enter element 6: 99
Enter element to search: 41
Index of 41 using linear search: -1
Index of 41 using binary search: -1
```

3.

```
Enter number of elements: 6
Enter element 1: 12
Enter element 2: 14
Enter element 3: 20
Enter element 4: 37
Enter element 5: 41
Enter element 6: 58
Enter element to search: 15
Index of 15 using linear search: -1
Index of 15 using binary search: -1
```

4.

```
Enter number of elements: 5
Enter element 1: 105
Enter element 2: 210
Enter element 3: 321
Enter element 4: 410
Enter element 5: 500
Enter element to search: 105
Index of 105 using linear search: 0
Index of 105 using binary search: 0
```

5.

```
Enter number of elements: 6
Enter element 1: 7
Enter element 2: 10
Enter element 3: 11
Enter element 4: 19
Enter element 5: 25
Enter element 6: 33
Enter element to search: 33
Index of 33 using linear search: 5
Index of 33 using binary search: 5
```

6.

```
Enter number of elements: 6
Enter element 1: 31
Enter element 2: 42
Enter element 3: 57
Enter element 4: 68
Enter element 5: 73
Enter element 6: 89
Enter element to search: 90
Index of 90 using linear search: -1
Index of 90 using binary search: -1
```

Conclusion: Hence, in this practical, we have studied what coding style is and how to write code using a specific coding style. We have also studied

and analysed the time complexity of linear search and binary search algorithms.