LABORATORY REPORT

Algorithm Laboratory (CS-39001)

B.Tech Program in ECSc

Submitted By

Name:- SANNIDHI DEB

Roll No: 2330044

Kalinga Institute of Industrial Technology (Deemed to be University) Bhubaneswar, India

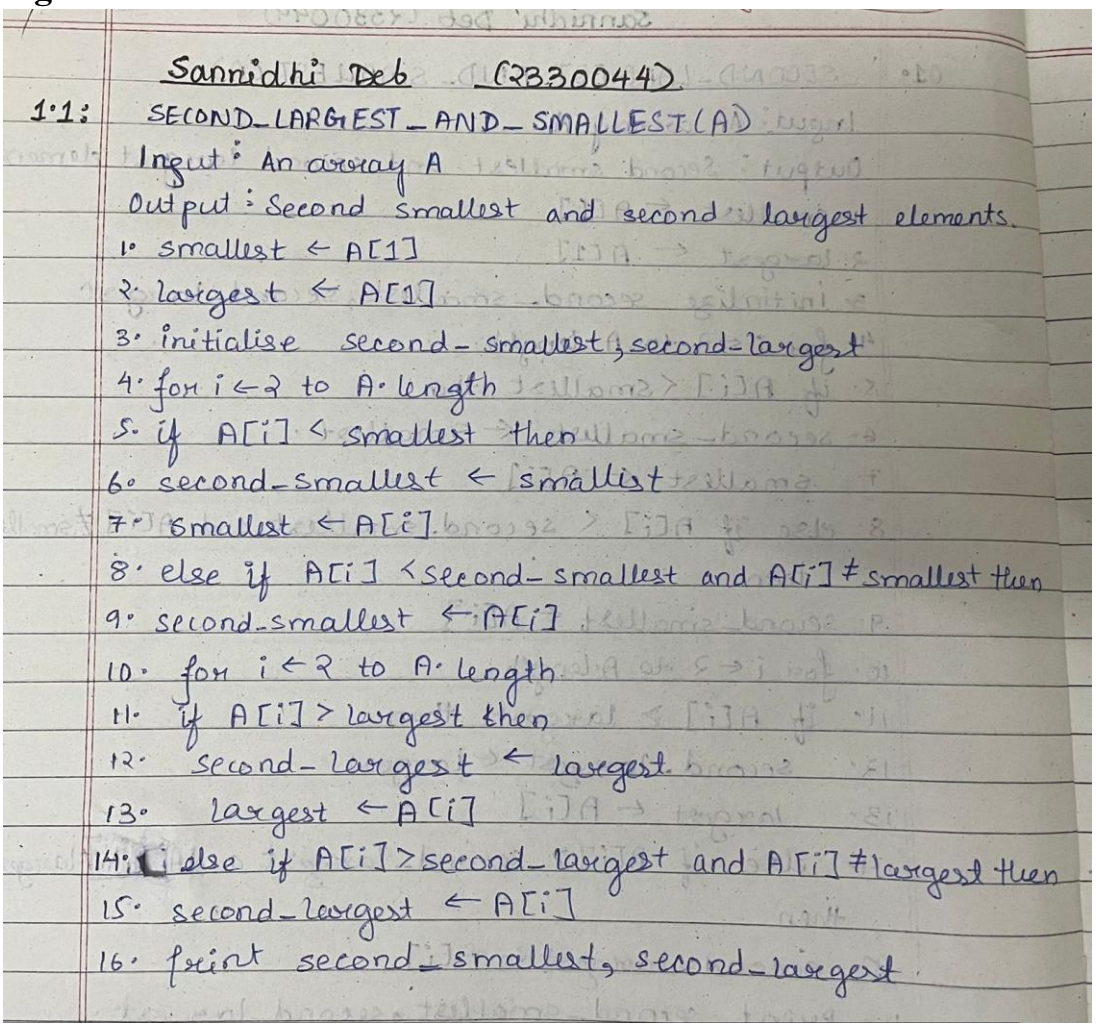
Autumn, 2025

Table of Contents

#	Title	Date of Experiment	Date of Submission	Remarks
1.	Revision of Data Structures	24/07/2025	06/08/2025	

Experiment Number	1.1
Experiment Title	Aim of the program: Write a program to find out the second smallest and second largest element stored in an array of n integers. Input: Size of the array is 'n' and read 'n' number of elements from a disc file. Output: Second smallest, Second largest
Date of Experiment	24/07/2025
Date of Submission	06/08/2025

1. Algorithm:-



Sannidhi Deb (2330044)

1.1: SECOND_LARGEST_AND_SMALLEST(A)

Input: An array A

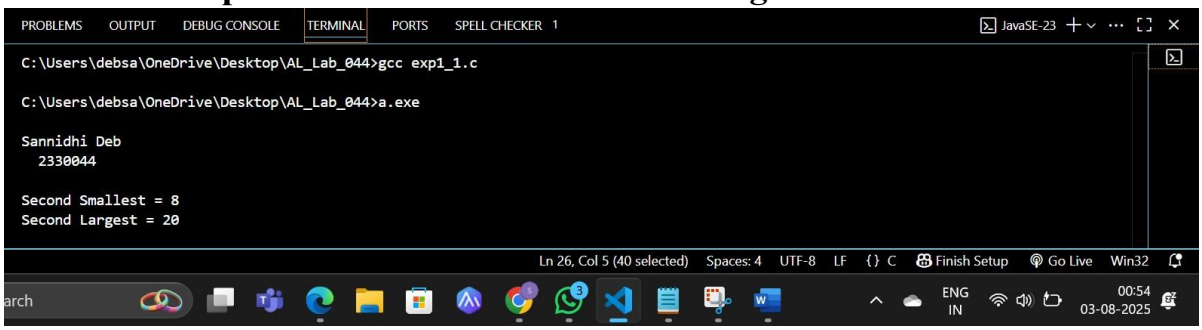
Output: Second smallest and second largest elements.

1. smallest ← A[1]
2. largest ← A[1]
3. initialise second_smallest, second_largest
4. for i ← 2 to A.length
5. if A[i] < smallest then
6. second_smallest ← smallest
7. smallest ← A[i]
8. else if A[i] < second_smallest and A[i] ≠ smallest then
9. second_smallest ← A[i]
10. for i ← 2 to A.length
11. if A[i] > largest then
12. second_largest ← largest
13. largest ← A[i]
14. else if A[i] > second_largest and A[i] ≠ largest then
15. second_largest ← A[i]
16. print second_smallest, second_largest

2. Code:-

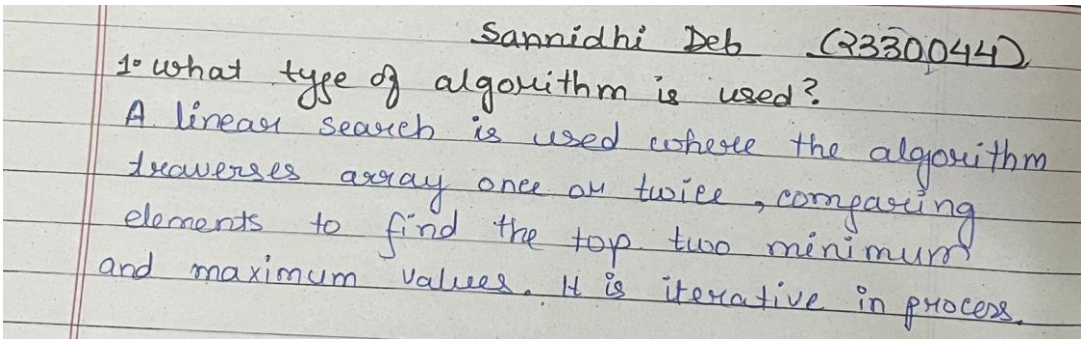
```
#include <stdio.h>
#include <limits.h>

void findSecondMinMax(int A[], int n) {
   int smallest = A[0], largest = A[0];
   int second_smallest = INT_MAX, second_largest = INT_MIN;

   for (int i = 1; i < n; i++) {        if
(A[i] < smallest) {
second_smallest = smallest;
          smallest = A[i];
      } else if (A[i] < second_smallest && A[i] != smallest) {
second_smallest = A[i];
      }
   }

   for (int i = 1; i < n; i++) {
if (A[i] > largest) {
second_largest = largest;
          largest = A[i];
      } else if (A[i] > second_largest && A[i] != largest) {
second_largest = A[i];
      }
   }

    printf("\nSannidhi Deb\n  2330044\n\n");

   printf("Second Smallest = %d\n", second_smallest);
   printf("Second Largest = %d\n", second_largest);
}

int main() {     int A[] = {10,
5, 20, 8, 30};
   int n = sizeof(A) / sizeof(A[0]);

   findSecondMinMax(A, n);

   return 0;
}
```

3. Results/Output:- Entire Screen Shot including Date & Time:-

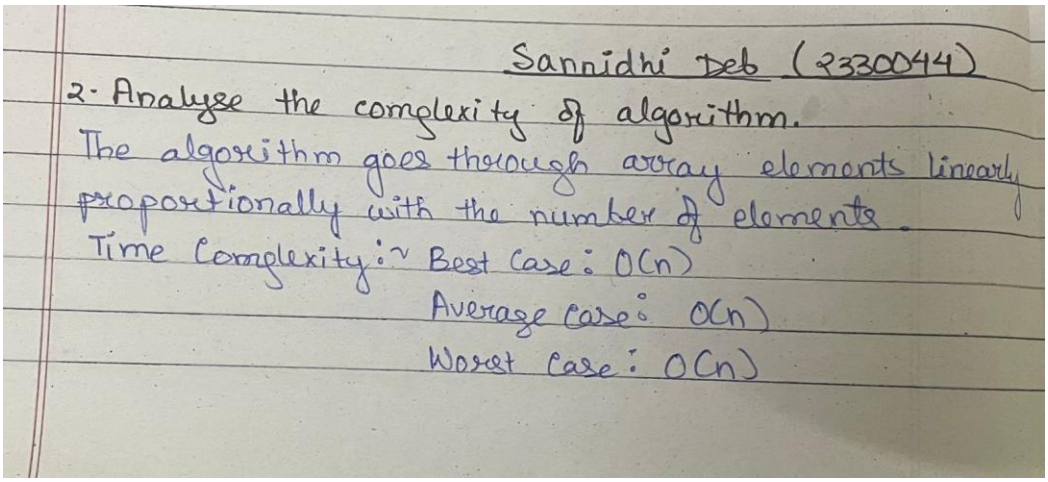


4. Remarks:-

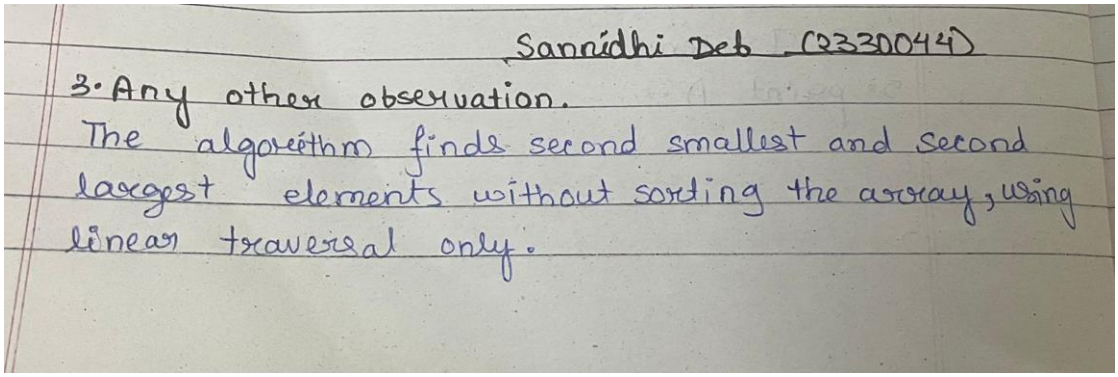
1. What type of algorithm is used?



2. Analyze the complexity of your algorithm.

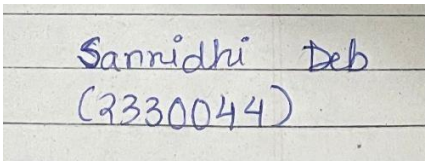


3.Any other observations?


Sannidhi Deb (2330044)

3. Any other observation.

The algorithm finds second smallest and second largest elements without sorting the array, using linear traversal only.


Sannidhi Deb
(2330044)

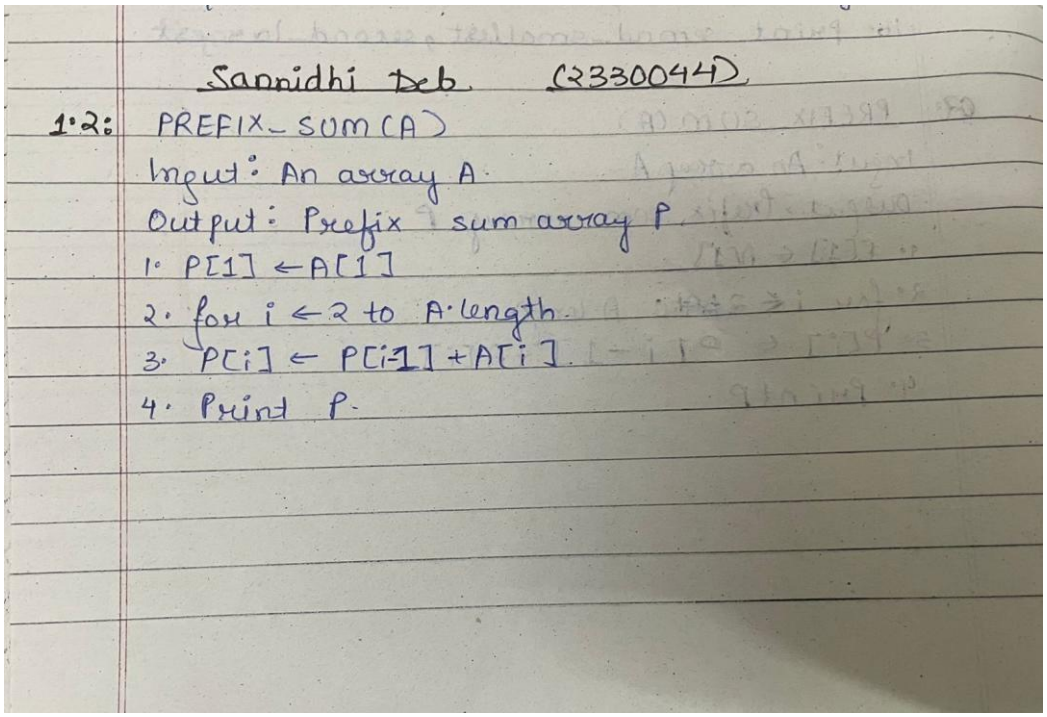
Signature of the FIC

Sannidhi Deb

(Name of the FIC)

Experiment Number	1.2
Experiment Title	Aim of the program: Given an array arr[] of size N, find the prefix sum of the array. A prefix sum array is another array prefixSum[] of the same size, such that the value of prefixSum[i] is arr[0] + arr[1] + arr[2] . . . arr[i]. Input Array: 3 4 5 1 2 Output Array: 3 7 12 13 15
Date of Experiment	24/07/2025
Date of Submission	06/08/2025

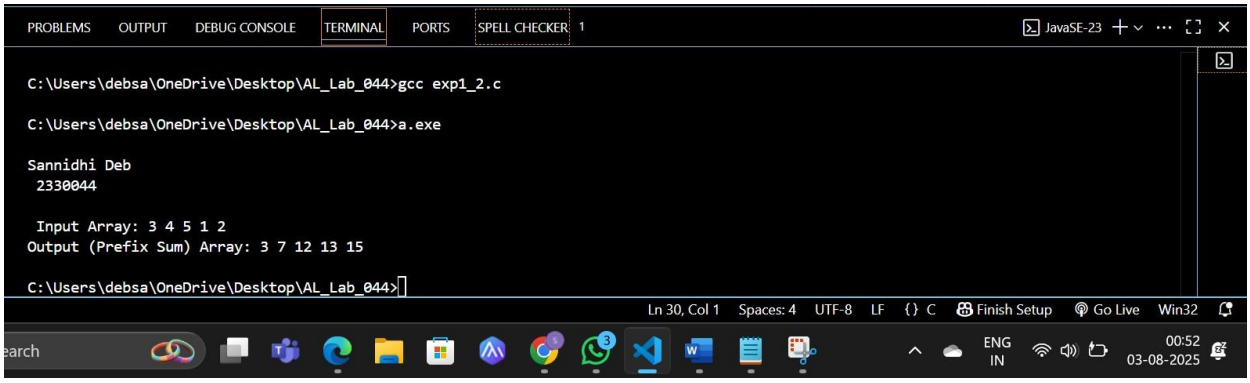
1. Algorithm:-



2. Code:-

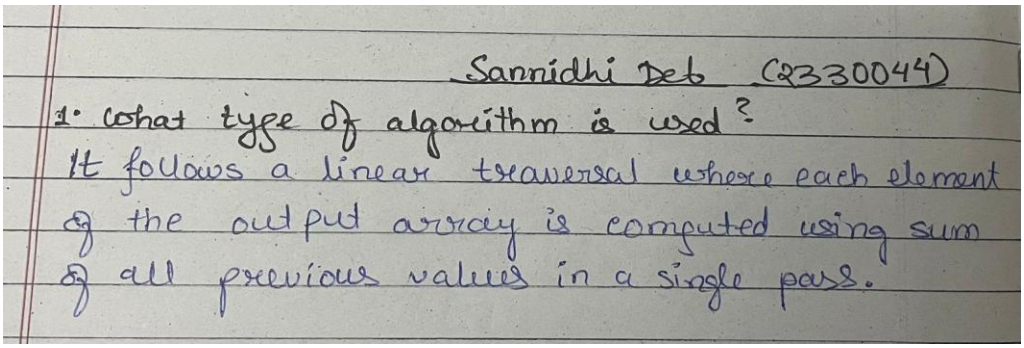
```
#include <stdio.h> void
prefixSum(int arr[], int n) {
int prefix[n];
   prefix[0] = arr[0];

   for(int i = 1; i < n; i++) {
      prefix[i] = prefix[i - 1] + arr[i];
   }
   printf("\nSannidhi Deb\n 2330044\n\n ");
   printf("Input Array: ");
for(int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }

   printf("\nOutput (Prefix Sum) Array: ");
for(int i = 0; i < n; i++) {
      printf("%d ", prefix[i]);
   }
   printf("\n");
}

int main() {     int arr[] =
{3, 4, 5, 1, 2};
   int n = sizeof(arr) / sizeof(arr[0]);

   prefixSum(arr, n);

   return 0;
}
```

3. Results/Output:- Entire Screen Shot including Date & Time:-

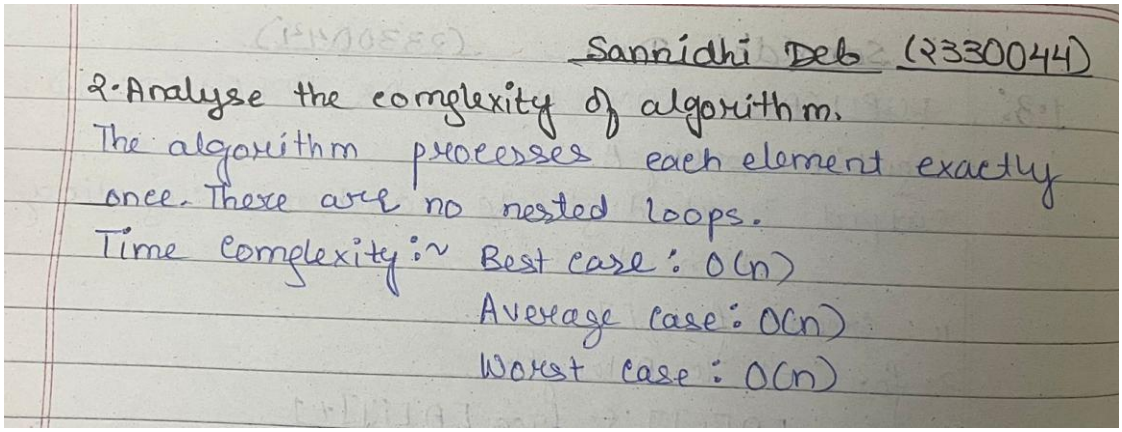


4. Remarks:-

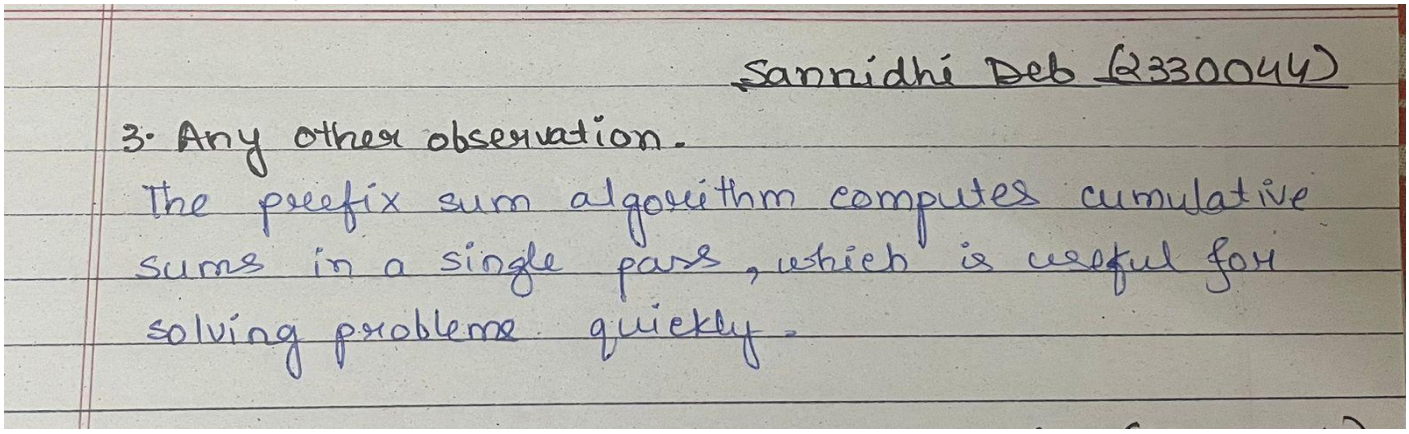
1. What type of algorithm is used?



2. Analyze the complexity of your algorithm.

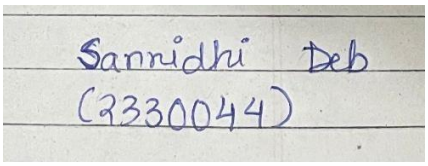


3.Any other observations?


Sannidhi Deb (2330044)

3. Any other observation.

The prefix sum algorithm computes cumulative sums in a single pass, which is useful for solving problems quickly.


Sannidhi Deb
(2330044)

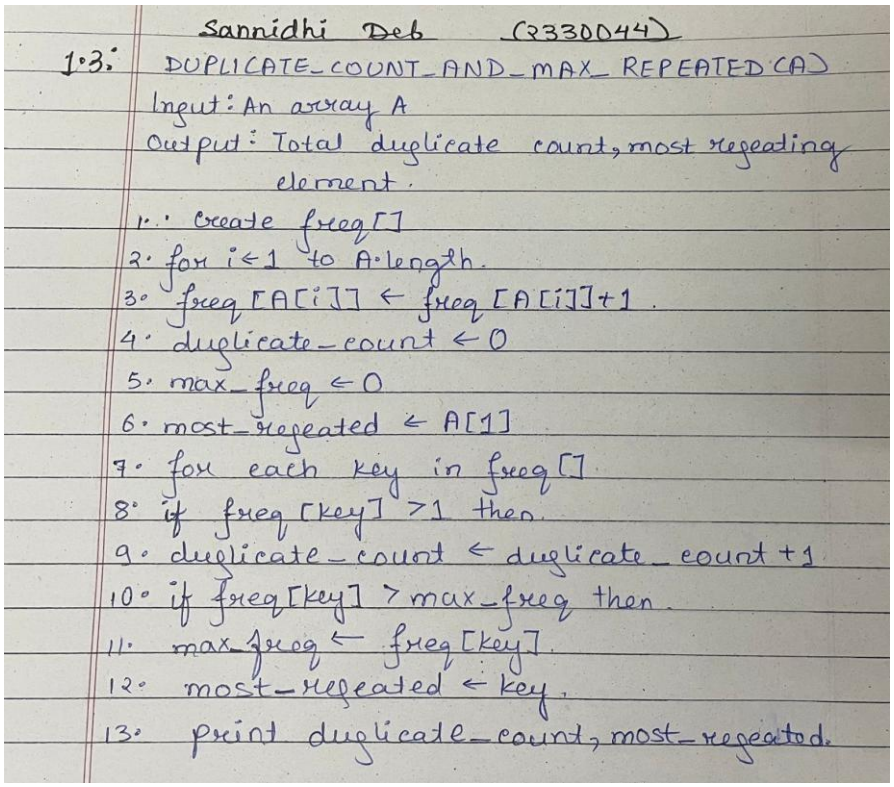
Signature of the FIC

Sannidhi Deb

(Name of the FIC)

Experiment Number	1.3
Experiment Title	Aim of the program: Write a program to read 'n' integers from a disc file that must contain some duplicate values and store them into an array. Perform the following operations on the array. a) Find out the total number of duplicate elements. b) Find out the most repeating element in the array. Input: Enter how many numbers you want to read from file: 15 Output: The content of the array: 10 40 35 47 68 22 40 10 98 10 50 35 68 40 10 Total number of duplicate values = 4 The most repeating element in the array = 10
Date of Experiment	31/07/2025
Date of Submission	06/08/2025

1. Algorithm:-



Sannidhi Deb (2330044)

1.3: DUPLICATE_COUNT_AND_MAX_REPEATED(A)

Input: An array A

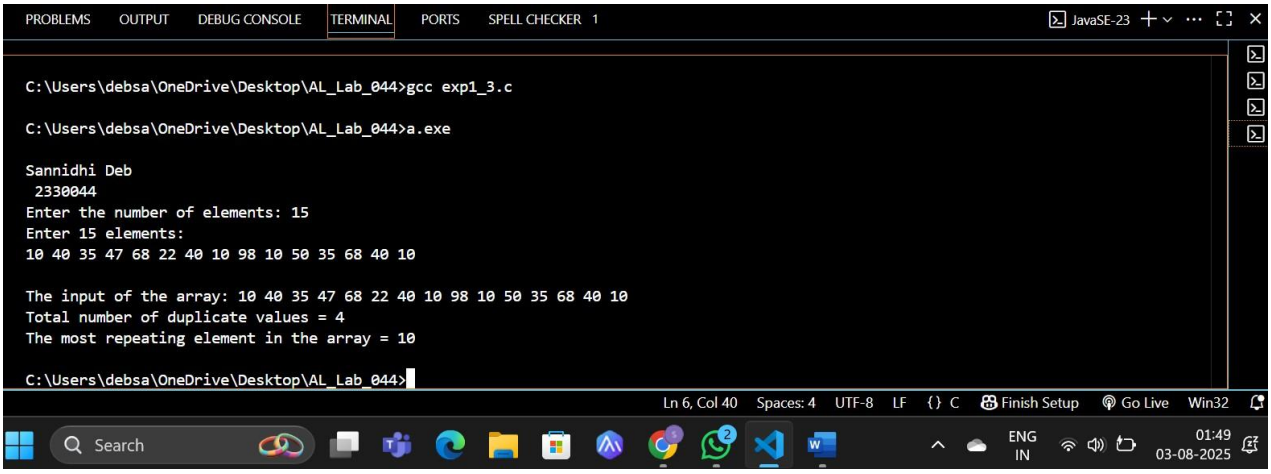
Output: Total duplicate count, most repeating element.

1. Create freq[]
2. for i ← 1 to A.length.
3. freq[A[i]] ← freq[A[i]]+1.
4. duplicate_count ← 0
5. max_freq ← 0
6. most_repeated ← A[1]
7. for each key in freq[]
8. if freq[key] > 1 then.
9. duplicate_count ← duplicate_count + 1
10. if freq[key] > max_freq then.
11. max_freq ← freq[key].
12. most_repeated ← key.
13. print duplicate_count, most_repeated.

2. Code:-

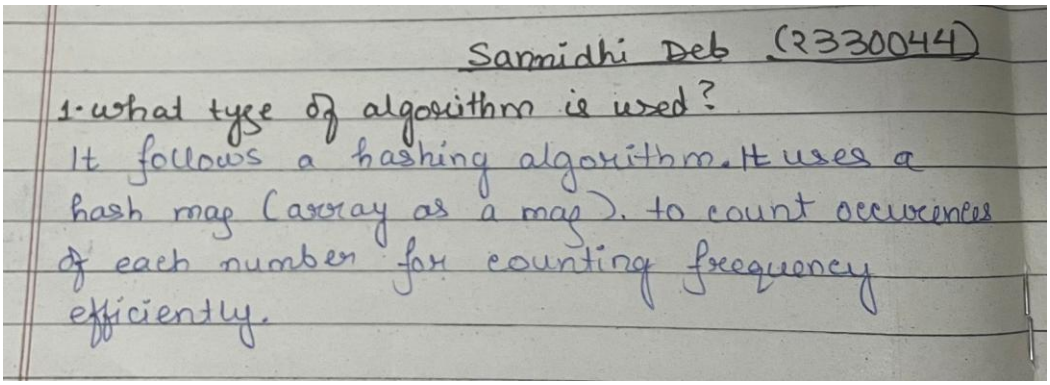
```
#include <stdio.h>
#define MAX 1000  int
main() {
   int A[100], freq[MAX] = {0};
   int n;
printf("\nSannidhi Deb\n 2330044\n\n");
printf("Enter the number of elements: ");
scanf("%d", &n);

   printf("Enter %d elements:\n", n);
for (int i = 0; i < n; i++) {
scanf("%d", &A[i]);
      freq[A[i]]++;
   }
int duplicate_count = 0;
int max_freq = 0;
   int most_repeated = A[0];

   for (int key = 0; key < MAX; key++) {
      if (freq[key] > 1) {
         duplicate_count++;
      }
      if (freq[key] > max_freq) {
max_freq = freq[key];
         most_repeated = key;
      }
   }
printf("\nThe input of the array: ");
for (int i = 0; i < n; i++) {
      printf("%d ", A[i]);
   }

   printf("\nTotal number of duplicate values = %d", duplicate_count);
printf("\nThe most repeating element in the array = %d\n", most_repeated);

   return 0;
}
```

3. Results/Output:- Entire Screen Shot including Date & Time:-

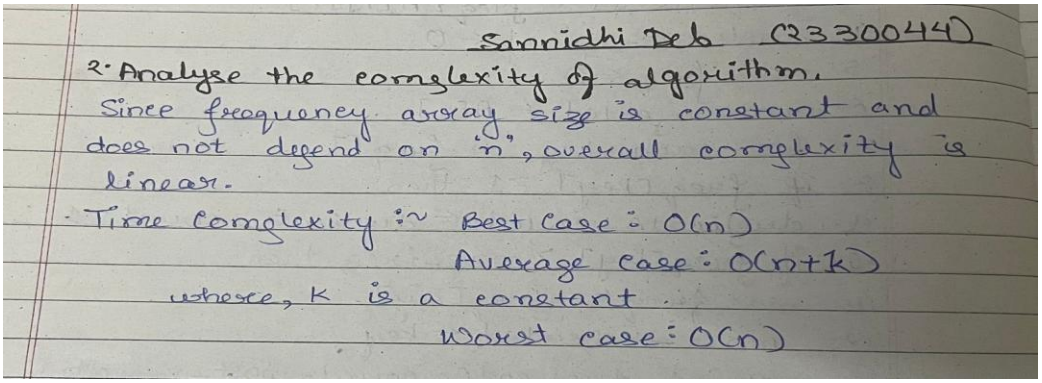


4. Remarks:-

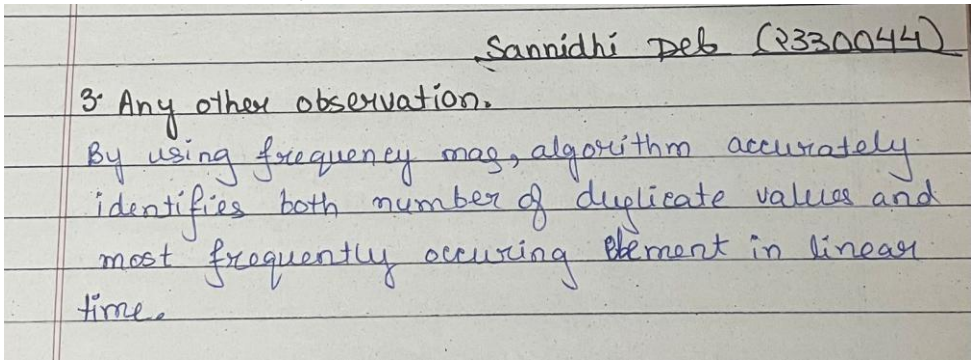
1. What type of algorithm is used?



2. Analyze the complexity of your algorithm.

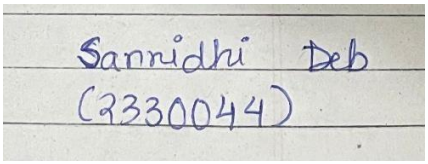


3.Any other observations?


Sannidhi Deb (2330044)

3. Any other observation.

By using frequency map, algorithm accurately identifies both number of duplicate values and most frequently occurring element in linear time.


Sannidhi Deb
(2330044)

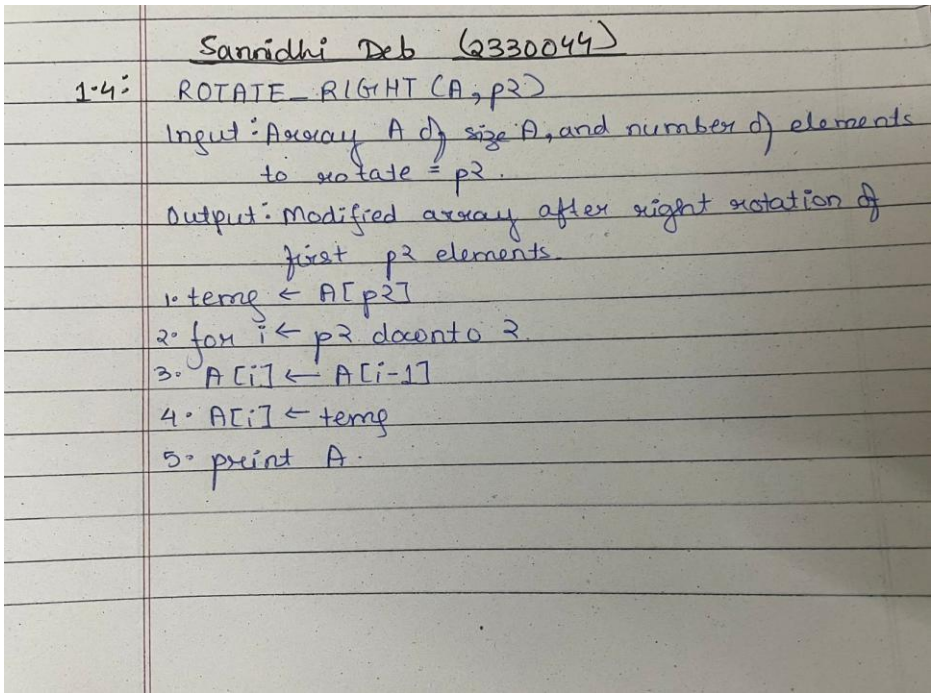
Signature of the FIC

Sannidhi Deb

(Name of the FIC)

Experiment Number	1.4
Experiment Title	Aim of the program: Write a function to ROTATE_RIGHT (p1, p2) right an array for first p2 elements by 1 position using EXCHANGE (p, q) function that swaps/exchanges the numbers p & q. Parameter p1 be the starting address of the array and p2 be the number of elements to be rotated. Input: Enter an array A of size N (9): 11 22 33 44 55 66 77 88 99 Call the function ROTATE_RIGHT (A, 5) Output: Before ROTATE: 11 22 33 44 55 66 77 88 99 After ROTATE: 55 11 22 33 44 66 77 88 99
Date of Experiment	01/07/2025
Date of Submission	06/08/2025

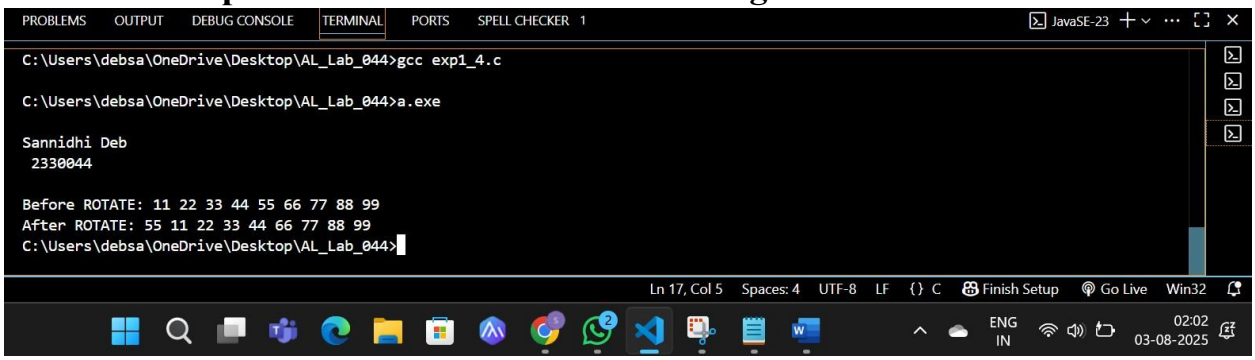
1. Algorithm:-



2. Code:-

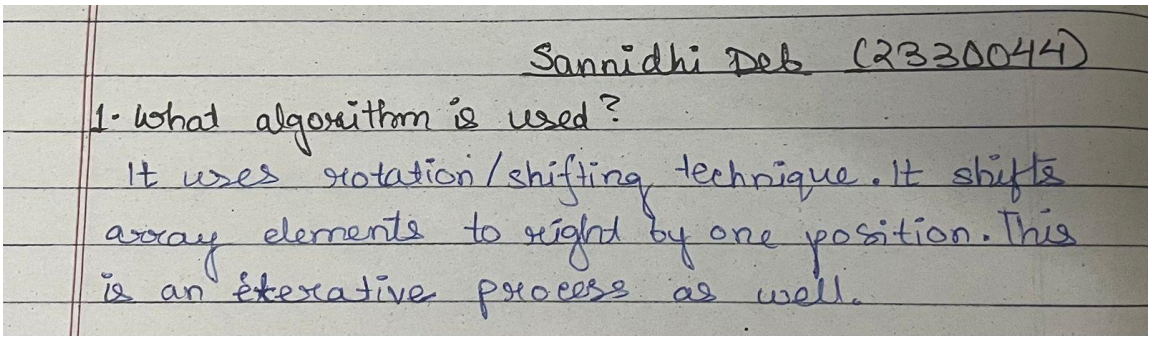
```
#include <stdio.h> void
EXCHANGE(int *p, int *q) {
int temp = *p;     *p = *q;
   *q = temp;
}

void ROTATE_RIGHT(int *p1, int p2) {
   for (int i = p2 - 1; i > 0; i--) {
      EXCHANGE(&p1[i], &p1[i - 1]);
   }
}

int main() {
   int A[9] = {11, 22, 33, 44, 55, 66, 77, 88, 99};
   int N = 9;

printf("\nSannidhi Deb\n 2330044\n\n");
printf("Before ROTATE: ");     for (int i = 0;
i < N; i++) {
      printf("%d ", A[i]);
   }

   ROTATE_RIGHT(A, 5);

   printf("\nAfter ROTATE: ");
for (int i = 0; i < N; i++) {
      printf("%d ", A[i]);
   }

   return 0;
}
```

3. Results/Output:- Entire Screen Shot including Date & Time:-

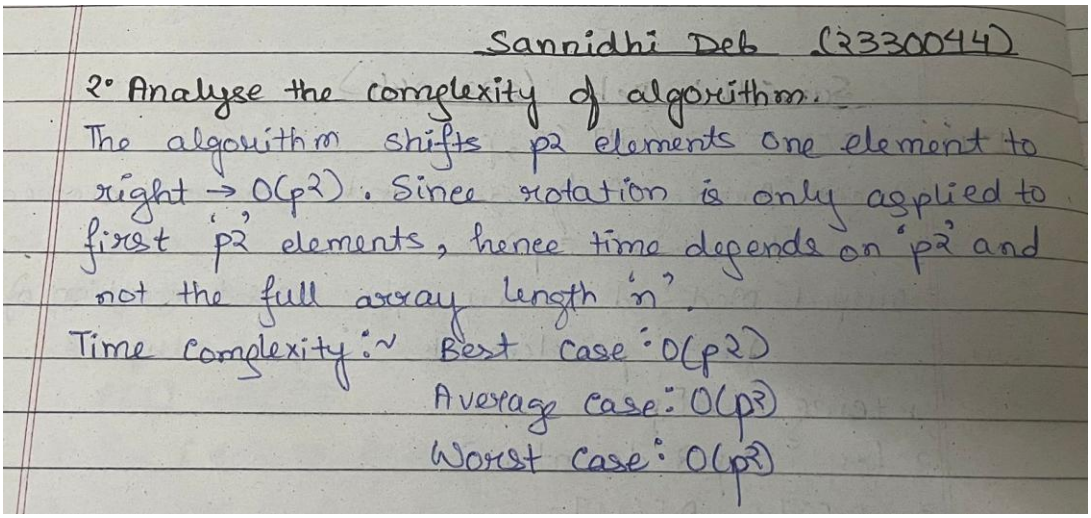


4. Remarks:-

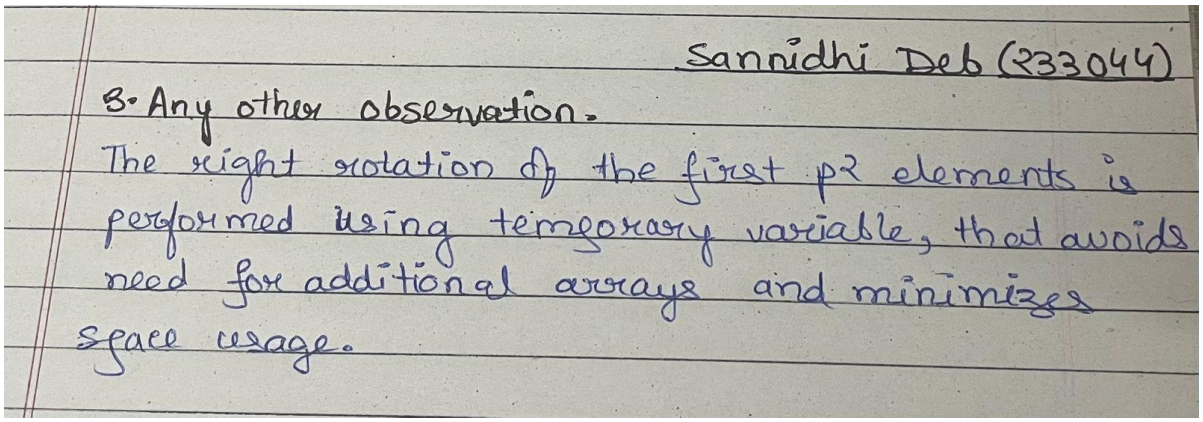
1. What type of algorithm is used?



2. Analyze the complexity of your algorithm.

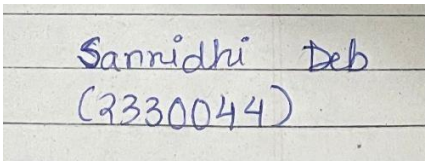


3.Any other observations?



5. Conclusion:-

In this lab session, we revised essential array-based operations in C programming. We implemented programs to find the second smallest and second largest elements, calculate prefix sums, detect duplicate values and identify the most frequent one, and perform partial array rotation using a swapping function. These exercises reinforced key data structure concepts such as array traversal, in-place updates, and efficient element manipulation — all fundamental for building more complex algorithms.



Signature of the FIC

Sannidhi Deb

(Name of the FIC)