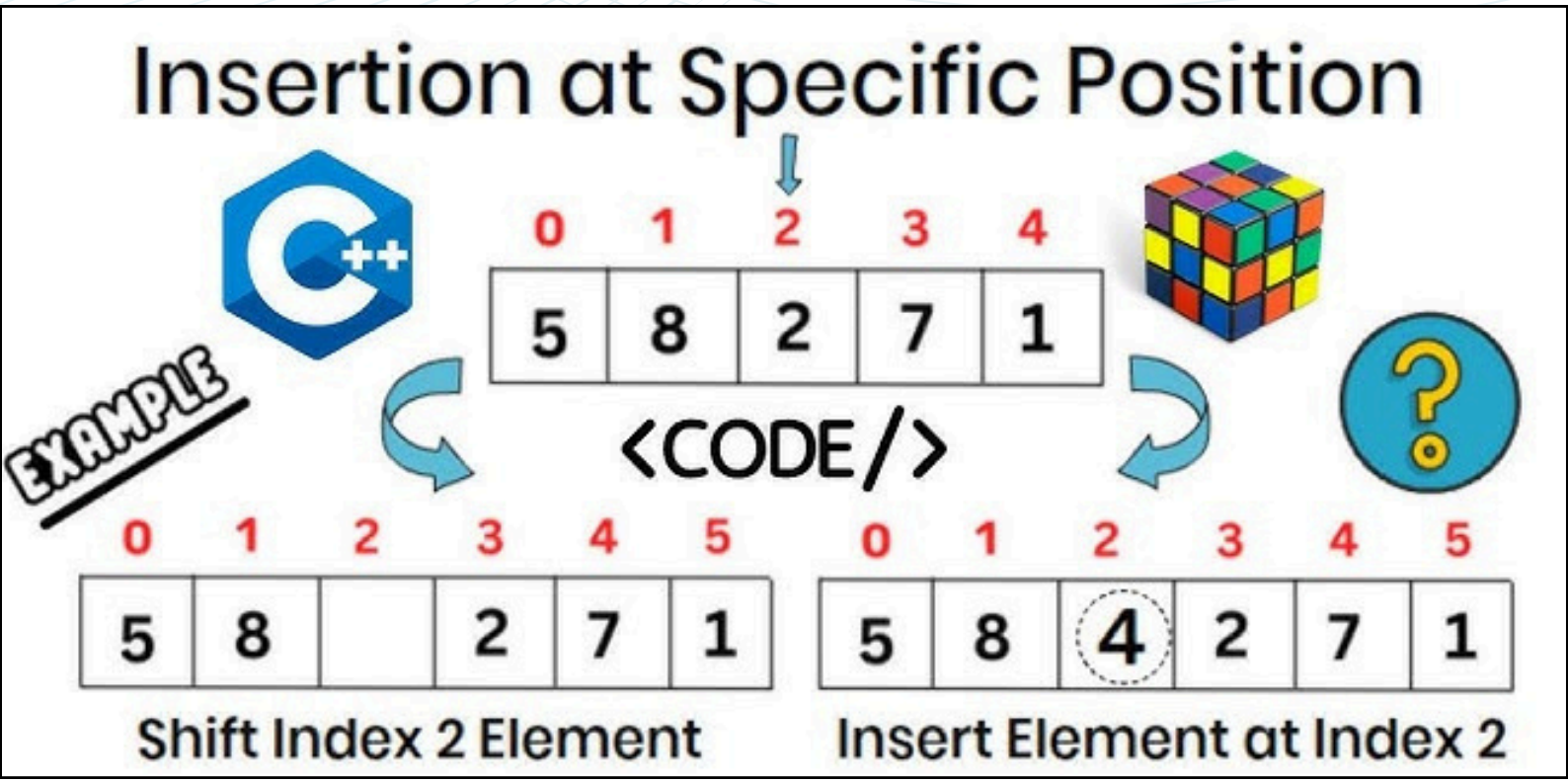
FAZAIA BILQUIS COLLEGE
NUR KHAN BASE,RWP



Task 1 – Introduction to Arrays

SUBMITTED TO: *MA'AM SANA*

ROLL NO: *BSCS-13-F24-01*

SUBJECT: *OOPS IN JAVA*

SUBMITTED BY: *MARIA ATTA*

INTRODUCTION TO ARRAYS

Definition:

- An array in C++ is a collection of elements of the same data type stored in contiguous memory locations. It allows storing multiple values under a single variable name and accessing them using an index.

TASK 1: ROLL-CALL REVERSE (INPUT + REVERSE TRAVERSAL)

Scenario:
You noted 5 student names in the order they arrived. You want to announce them in reverse for a fun activity.

- **Program:**

program 1.cpp

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

int main() {
    string names[5];
    cout << "Enter the 5 names (one word each): ";
    for (int i = 0; i < 5; i++) {
        cin >> names[i];
    }
    // Print in reverse
    cout << "Reverse Order: ";
    for (int i = 4; i >= 0; i--) {
        cout << names[i];
        if (i > 0) cout << " ";
    }
    cout << "\n";
    return 0;
}
```

- **Output:**

```
C:\Users\LAB-1\Desktop\DS L
Enter the 5 names (one word each): Maham
Noor
Zuni
Mahrukh
Esha
Reverse Order: Esha Mahrukh Zuni Noor Maham
```

- **Code Explanation:**
  - We declared an array **arr** of size 5 and initialized it with values.
  - A ***for loop*** is used to iterate through the array.
  - The program prints each element of the array one by one.

TASK 2: MINI TEMPERATURE STATS (MIN, MAX, AVERAGE)

Scenario: A sensor saved temperatures for 7 hours. Find the lowest, highest, and average.

- **Program:**

Program2.cpp

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    int t[7]; // 7 reading
    cout<< "Enter the 7 integers temperature: ";
    for(int i = 0; i < 7; i++){
        cin>> t[i];
    }
//Start with the first value as both min and max
    int mn = t[0];
    int mx = t[0];
    int sum = 0;
//   One pass to compute min max and sum
for(int i=0; i<7; i++){
    if(t[i] < mn) mn = t[i];
    if(t[i] > mx) mx = t[i];
    sum += t[i];
}
    double avg = sum / 7.0;
    cout<<"Min = "<< mn << ",Max = " << mx <<",Avg = " << fixed << setprecision(2) << avg <<"\n";
    return 0;
}
```

- **Output:**

```
C:\Users\LAB-1\Desktop\DS L
Enter the 7 integers temperature: 98
45
56
78
23
21
12
Min = 12,Max = 98,Avg = 47.57
```

- **Code Explanation:**
  - An array **t[7]** stores 7 temperature readings entered by the user.
  - We initialize **mn** and **mx** with the first value.
  - A single loop calculates the ***min***, ***max***, and also adds values for the sum.
  - The average is found by dividing ***sum*** by **7.0** (to get decimal result).

TASK 3 - LOST & FOUND SEARCH (LINEAR SEARCH + COMPARISONS)

Scenario: Security logged item IDs found on campus. A student asks if a specific ID exists. Search one by one and report the index and number of comparisons.

- **Program:**

Program3.cpp

```cpp
#include<iostream>
using namespace std;
int main(){
    const int CAP = 100; // capacity
    int a[CAP];
    int n;
    cout<<"How many IDs? ";
    cin>>n;
    if(n < 0 || n > CAP){
        cout<<"Invalid size. \n";
        return 0;
    }
    cout<<"Enter "<<n<<" IDs: ";
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    int target;
    cout<<"Serach ID: ";
    cin >> target;

    int index = -1;
    int comparisons = 0;
    //  Linear Search
    for(int i = 0; i < n; i++){
        comparisons++;
        if(a[i] == target){
            index = i;
            break; //stop at first match
        }
    }
    if(index != -1){
        cout<<"Found at index " << index << " (comparisons: " << comparisons << " )\n";
    }else{
        cout<<"Not , Found (comparisons: " << comparisons << " ) \n";
    }
    return 0;
}
```

- **Output:**

```
C:\Users\LAB-1\Desktop\DS L
How many IDs? 7
Enter 7 IDs: 101
102
103
78
56
76
12
Serach ID: 101
Found at index 0 (comparisons: 1 )
```

- **Code Explanation:**
  - The program first takes **n** IDs from the user.
  - The **linear search** loop compares each ID with the target.
  - If found, it reports the index and how many comparisons were made.
  - If not found, it shows ***"Not found"*** along with the total comparisons.

TASK 4 - LIBRARY WAITLIST (INSERT AT POSITION WITH RIGHT SHIFT)

Scenario: Insert a new member ID at a given position in the library waitlist by shifting existing elements to the right.

- **Program:**

Program4.cpp

```cpp
#include<iostream>
using namespace std;
int main(){
    const int CAP = 30; // capacity
    int a[CAP];
    int n;
    cout<<"Initial size (0.."<< CAP << ")";
    cin>>n;
    if(n < 0 || n > CAP){
        cout<<"Invalid size. \n";
        return 0;
    }
    cout<<"Enter "<<n<<" member IDs: ";
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    int pos, val;
    cout<<"Insert Position (0.." << n <<"): ";
    cin >> pos;
    cout<<"Value to insert:";
    cin >> val;

    //  Basic check capacity and valid position
    if(n == CAP || pos < 0 || pos > n){
        cout<<"Insert Failed (overflow or bad position). \n";
        return 0;
    }

    //  Shift right from the end toward pos
    for(int i = n; i > pos; i--){
        a[i] = a[i - 1];
    }

    a[pos] - val;
    n++; //size incresed
    cout<<"After insert: ";
    for(int i = 0; i < n; i++){
        cout<<a[i]<<(i + 1<n?" ":"\n");
    }
    return 0;
}
```

- **Output:**

```
C:\Users\LAB-1\Desktop\DS L
Initial size (0..30)5
Enter 5 member IDs: 101
12
23
34
45
Insert Position (0..5): 2
Value to insert:23
After insert: 101 12 23 23 34 45
```

- **Code Explanation:**
  - The program first reads the initial size and the member IDs.
  - The user specifies the position and the new value to insert.
  - Elements from the end are shifted right one step to free the position.
  - The new value is inserted, the size increases by 1, and the updated array is printed.

TASK 5: CANTEEN TOKENS – DELETE AT INDEX (LEFT SHIFT)

Scenario: The canteen queue is stored as token numbers in an array. When the student at index k is served, remove that token and shift left to close the gap.

- **Program:**

Source1.cpp

Source1.cpp > main()

```cpp
#include <iostream>
using namespace std;
int main()
{
    const int CAP = 100;
    int a[CAP]; // array of integers
    int n;
    cout << "Size: ";
    cin >> n;
    if (n < 0 || n > CAP){
        cout << "Invalid size" << endl;
        return 0;
    }cout << "Enter " << n << " tokens: "; // read n integers
    for (int i = 0; i < n; i++){
        cin >> a[i]; // read an integer
    }
    int k;
    cout << "Delete index (0.." << n - 1 << "): ";
    cin >> k;
    if (k < 0 || k >= n){
        cout << "Invalid index" << endl;
        return 0;
    }
    for (int i = k; i < n - 1; i++){// Shift Left from k
        a[i] = a[i + 1]; // copy from right to left
    } n--; // reduce size by 1
    cout << "After Delete: ";
    for (int i = 0; i < n; i++){
        cout << a[i] << (i + 1 < n ? ' ' : '\n');
    }
    return 0;
}
```

- **Output:**

Code

```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\" ; if ($?) { g++ Source1.cpp -o Source1 } ; if ($?) { .\Source1 }
Size: 4
Enter 4 tokens: 23
34
45
56
Delete index (0..3): 2
After Delete: 23 34 56
PS C:\Users\manan\Desktop\DSA Lab Tasks>
```

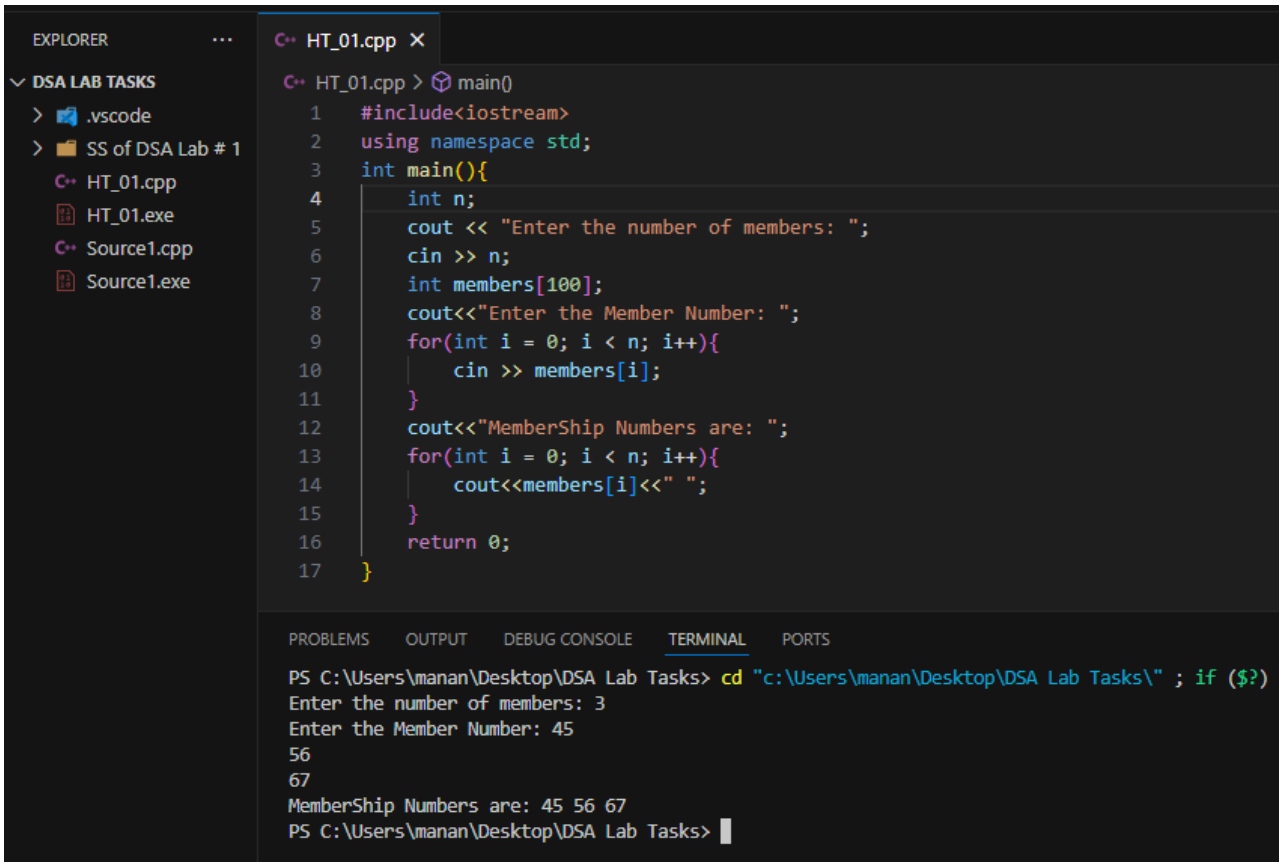
- **Code Explanation:**
  - Takes **n** numbers into an array.
  - Asks for an index **k** to delete.
  - Shifts all elements after **k** one step left.
  - Reduces size by **1**.
  - Prints the updated array.

HOME TASKS

PART A – EXTRA BASIC ARRAY PRACTICE (QUICK WARM-UP)

1. Read & Show (Members): Read N membership numbers into an array and print them on one line.

- **Program:**



- **Code Explanation:**
  - **cin >> n;** → takes size of members list.
  - Loop stores **n** numbers in **members[].**
  - Another loop prints all members.

2. Make a Copy (Members): Copy the membership array into another array and print the copy to confirm.

- **Program:**

HT_02.cpp

```cpp
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of members: ";
    cin >> n;
    int members[100], copy[100];
    cout << "Enter membership numbers: ";
    for (int i = 0; i < n; i++) {
        cin >> members[i];
    }
    for (int i = 0; i < n; i++) {
        copy[i] = members[i];
    }
    cout << "Copied array: ";
    for (int i = 0; i < n; i++) {
        cout << copy[i] << " ";
    }
    return 0;
}
```

```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\" ; if ($?) { g++ HT_02.cpp -o HT_02 }
Enter number of members: 3
Enter membership numbers: 34
45
76
Copied array: 34 45 76
PS C:\Users\manan\Desktop\DSA Lab Tasks>
```

- **Code Explanation:**
  - Input size **n** and fill **members[]**.
  - Loop copies each value into **copy[]**.
  - Print the **copy[]** array.

3. Sum & Count (Visitors): Read M hourly visitor counts (non-negative) and print the total visitors and the count of hours recorded.

- **Program:**

HT_03.cpp

```cpp
#include <iostream>
using namespace std;
int main() {
    int m;
    cout << "Enter number of hours: ";
    cin >> m;
    int visitors[100];
    cout << "Enter visitors for each hour: ";
    for (int i = 0; i < m; i++) {
        cin >> visitors[i];
    }
    int sum = 0;
    for (int i = 0; i < m; i++) {
```

```
        sum += visitors[i];   // add all visitors
    }
    cout << "Total visitors = " << sum << endl;
    cout << "Count of hours = " << m << endl;
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\" ; if ($?) { g++ HT_03.cpp -o HT_03 }
Enter number of hours: 4
Enter visitors for each hour: 6
9
12
24
Total visitors = 51
Count of hours = 4
PS C:\Users\manan\Desktop\DSA Lab Tasks>
```

- **Code Explanation:**
  - Input **n** (hours) and **visitors[]**.
  - Loop adds all visitors to **sum**.
  - Print total visitors and hours count.

4. First & Last Match (Members): Ask for a membership number. Print the first index where it appears and the last index where it appears (or -1 if not found).

- **Program:**

HT_04.cpp

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of members: ";
    cin >> n;
    int members[100];
    cout << "Enter membership numbers: ";
    for (int i = 0; i < n; i++) {
        cin >> members[i];
    }
    int target;
    cout << "Enter number to search: ";
    cin >> target;
    int first = -1, last = -1;
    for (int i = 0; i < n; i++) {
        if (members[i] == target) {
            if (first == -1) {
                first = i;   // set first time
            }
            last = i;        // update every time found
        }
    }
    cout << "First index = " << first << endl;
    cout << "Last index = " << last << endl;
    return 0;
}
```

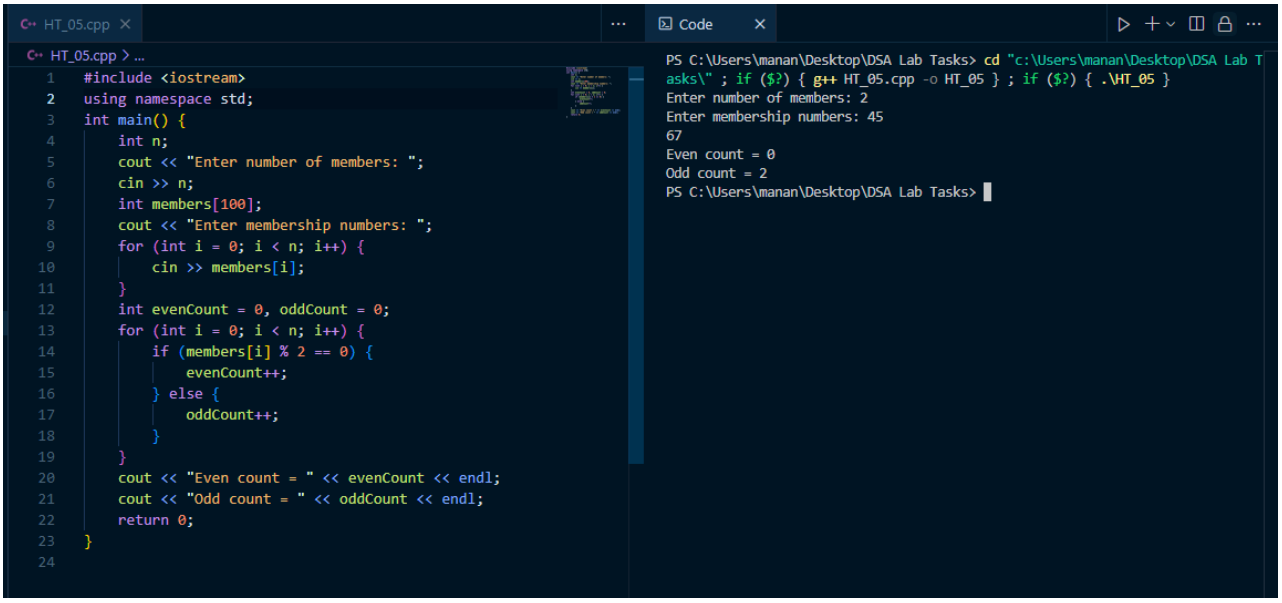
Code

```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\" ; if ($?) { g++ HT_04.cpp -o HT_04 } ; if ($?) { .\HT_04 }
Enter number of members: 4
Enter membership numbers: 23
34
56
67
Enter number to search: 56
First index = 2
Last index = 2
PS C:\Users\manan\Desktop\DSA Lab Tasks>
```

- **Code Explanation:**
  - Input **n**, array, and target.
  - Loop finds first index **(if target == arr[i])**.
  - Another loop finds last index.
  - Print results (or **-1** if not found).

5. Even/Odd Count (Members): Print how many membership numbers are even and how many are odd.

- **Program:**



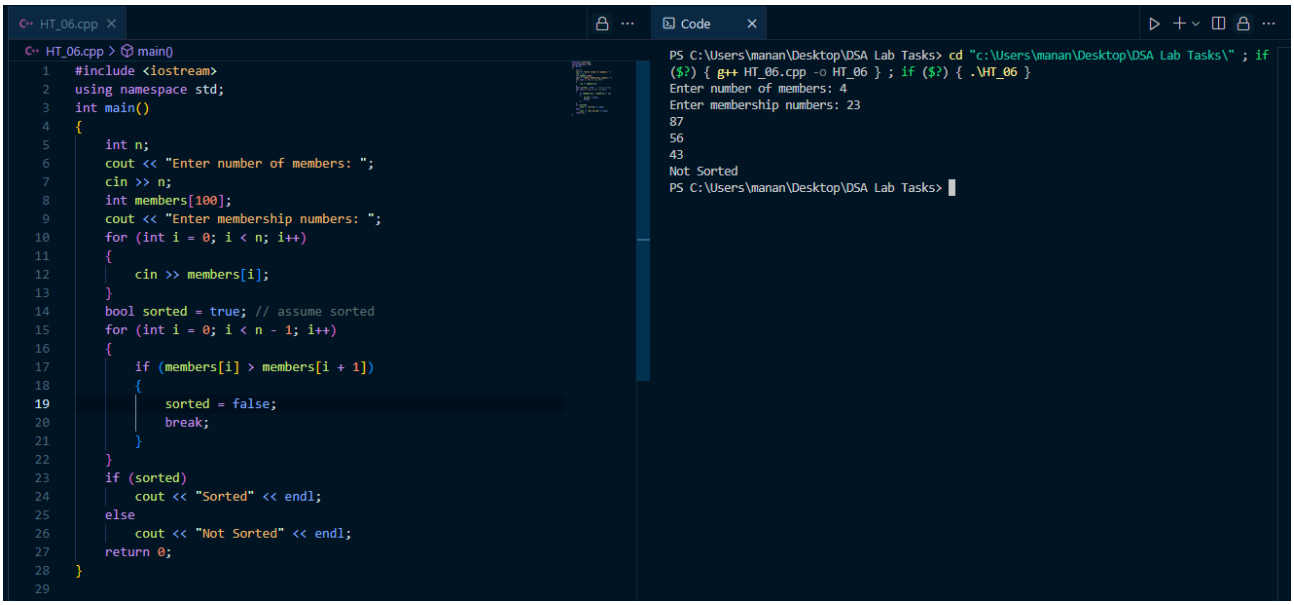
```cpp
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of members: ";
    cin >> n;
    int members[100];
    cout << "Enter membership numbers: ";
    for (int i = 0; i < n; i++) {
        cin >> members[i];
    }
    int evenCount = 0, oddCount = 0;
    for (int i = 0; i < n; i++) {
        if (members[i] % 2 == 0) {
            evenCount++;
        } else {
            oddCount++;
        }
    }
    cout << "Even count = " << evenCount << endl;
    cout << "Odd count = " << oddCount << endl;
    return 0;
}
```

```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\" ; if ($?) { g++ HT_05.cpp -o HT_05 } ; if ($?) { .\HT_05 }
Enter number of members: 2
Enter membership numbers: 45
67
Even count = 0
Odd count = 2
PS C:\Users\manan\Desktop\DSA Lab Tasks>
```

- **Code Explanation:**
  - Input size **n** and **members[]**.
  - Loop checks each number with **%2**.
  - Count evens and odds separately.
  - Print both counts.

6. Is Sorted? (Members): Check if the membership list is in non-decreasing order; print Sorted or Not Sorted.

- **Program:**



```cpp
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout << "Enter number of members: ";
    cin >> n;
    int members[100];
    cout << "Enter membership numbers: ";
    for (int i = 0; i < n; i++)
    {
        cin >> members[i];
    }
    bool sorted = true; // assume sorted
    for (int i = 0; i < n - 1; i++)
    {
        if (members[i] > members[i + 1])
        {
            sorted = false;
            break;
        }
    }
    if (sorted)
        cout << "Sorted" << endl;
    else
        cout << "Not Sorted" << endl;
    return 0;
}
```

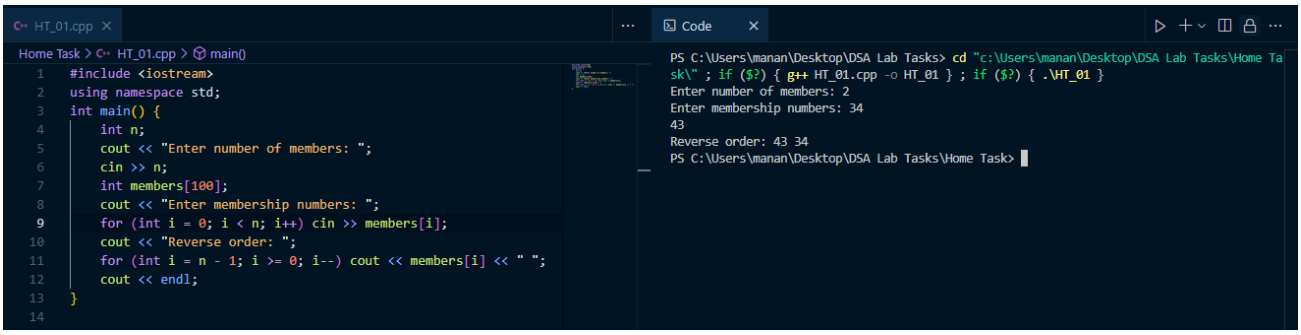
```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\" ; if ($?) { g++ HT_06.cpp -o HT_06 } ; if ($?) { .\HT_06 }
Enter number of members: 4
Enter membership numbers: 23
87
56
43
Not Sorted
PS C:\Users\manan\Desktop\DSA Lab Tasks>
```

- ## Code Explanation:
  - Input **n** and **members[]**.
  - Loop checks if **arr[i] > arr[i+1]**.
  - If yes → set sorted = **false**.
  - Print "**Sorted**" or "**Not Sorted**"

PART B – LIBRARY FEATURES (CORE SCENARIO TASKS)

1. Reverse Order (Membership Numbers): Print the membership numbers from last to first so the librarian can call students in reverse. (Do not permanently change the original list.)

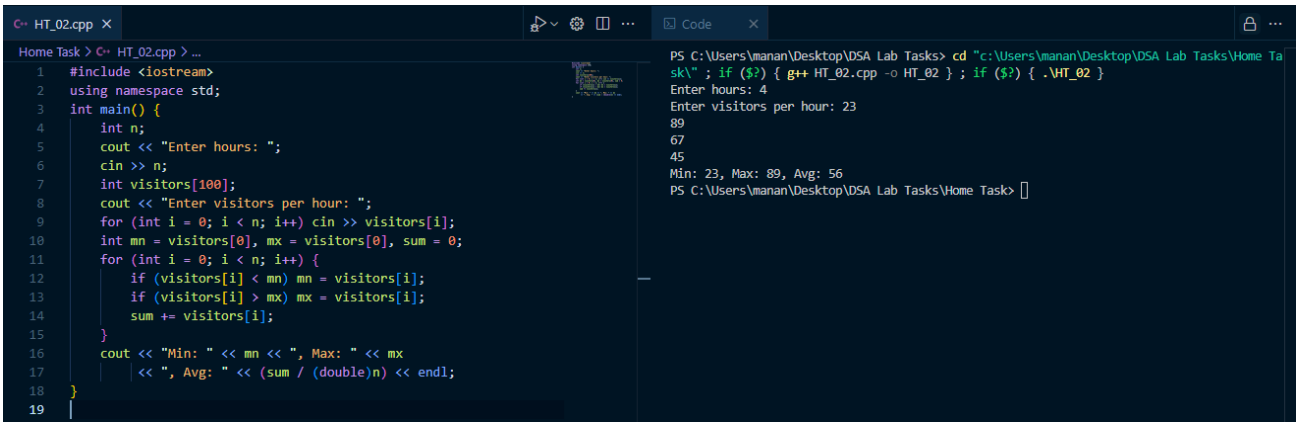
- ## Program:



- ## Code Explanation:
  - **for (int i = n - 1; i >= 0; i--) cout << members[i] << " ";**
  - Reads membership numbers into an array.
  - Prints them starting from the last index down to 0.
  - Does not modify the original array.

2. Quick Stats in One Pass (Hourly Visitor Count): From the visitor list, compute and print Min, Max, and Average. Do it in one loop by keeping a running min, max, and sum.

- ## Program:



- ## Code Explanation:
  - **if (visitors[i] < mn) mn = visitors[i];**
  - **if (visitors[i] > mx) mx = visitors[i];**
  - **sum += visitors[i];**
  - Loops once through the visitor list.
  - Tracks minimum, maximum, and running sum together.
  - Computes average using total **sum ÷ n.**

---

3. Linear Search with Comparison Count (Membership Numbers): Ask for a target membership number and search linearly. Show whether it's found, the index, and how many comparisons were made during the search.

- ## Program:

```
HT_03.cpp
Home Task > HT_03.cpp > main()
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of members: ";
    cin >> n;
    int members[100];
    cout << "Enter membership numbers: ";
    for (int i = 0; i < n; i++) cin >> members[i];
    int target, comparisons = 0;
    cout << "Enter number to search: ";
    cin >> target;
    int index = -1;
    for (int i = 0; i < n; i++) {
        comparisons++;
        if (members[i] == target) {
            index = i;
            break;
        }
    }
    if (index != -1)
        cout << "Found at index " << index;
    else
        cout << "Not found";
    cout << " | Comparisons: " << comparisons << endl;
}
```

```
Code
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\Home Task\" ; if ($?) { g++ HT_03.cpp -o HT_03 } ; if ($?) { .\HT_03 }
Enter number of members: 3
Enter membership numbers: 23
34
45
Enter number to search: 45
Found at index 2 | Comparisons: 3
PS C:\Users\manan\Desktop\DSA Lab Tasks\Home Task>
```

- ## Code Explanation:
  - **for (int i = 0; i < n; i++) {**
  - **comparisons++;**
  - **if (members[i] == target) { index = i; break; }}**
  - Compares each element with the target.
  - Stops immediately when found.
  - Reports index and number of comparisons.

---

4. Insert at a Given Position (Membership Numbers): A new member must appear at a particular position in the list. Insert the number at that position by shifting elements to the right, then show the updated list.

- ## Program:

```cpp
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of members: ";
    cin >> n;
    int members[100];
    cout << "Enter membership numbers: ";
    for (int i = 0; i < n; i++) cin >> members[i];
    int pos, newMem;
    cout << "Enter position and new member: ";
    cin >> pos >> newMem;
    if (pos < 0 || pos > n) {
        cout << "Invalid position.\n";
        return 0;
    }
    for (int i = n; i > pos; i--) {
        members[i] = members[i - 1];
    }
    members[pos] = newMem;
    n++;
    cout << "After insert: ";
    for (int i = 0; i < n; i++) cout << members[i] << " ";
    cout << endl;
}
```

```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\Home Task\" ; if ($?) { g++ HT_04.cpp -o HT_04 } ; if ($?) { .\HT_04 }
Enter number of members: 3
Enter membership numbers: 12
32
65
Enter position and new member: 2
32
After insert: 12 32 32 65
PS C:\Users\manan\Desktop\DSA Lab Tasks\Home Task>
```

- ## Code Explanation:
  - **for (int i = n; i > pos; i--) members[i] = members[i - 1];**
  - **members[pos] = newMem; n++;**
  - Shifts all elements from position to the right.
  - Places the new value at the desired position.
  - Increases array size by 1.

5. Delete at a Given Index (Membership Numbers): A wrong entry needs to be removed from a specific index. Delete it by shifting elements to the left, then show the updated list.

- ## Program:

```cpp
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of members: ";
    cin >> n;
    int members[100];
    cout << "Enter membership numbers: ";
    for (int i = 0; i < n; i++) cin >> members[i];
    int idx;
    cout << "Enter index to delete: ";
    cin >> idx;
    if (idx < 0 || idx >= n) {
        cout << "Invalid index.\n";
        return 0;
    }
    for (int i = idx; i < n - 1; i++) {
        members[i] = members[i + 1];
    }
    n--;
    cout << "After delete: ";
    for (int i = 0; i < n; i++) cout << members[i] << " ";
    cout << endl;
}
```

```
PS C:\Users\manan\Desktop\DSA Lab Tasks> cd "c:\Users\manan\Desktop\DSA Lab Tasks\Home Task\" ; if ($?) { g++ HT_05.cpp -o HT_05 } ; if ($?) { .\HT_05 }
Enter number of members: 3
Enter membership numbers: 12
23
34
Enter index to delete: 1
After delete: 12 34
PS C:\Users\manan\Desktop\DSA Lab Tasks\Home Task>
```

- ## Code Explanation:
  - **for (int i = idx; i < n - 1; i++) members[i] = members[i + 1];**
  - **n--;**
  - Shifts elements left from the delete index.
  - Overwrites the deleted value.
  - Reduces array size by 1.