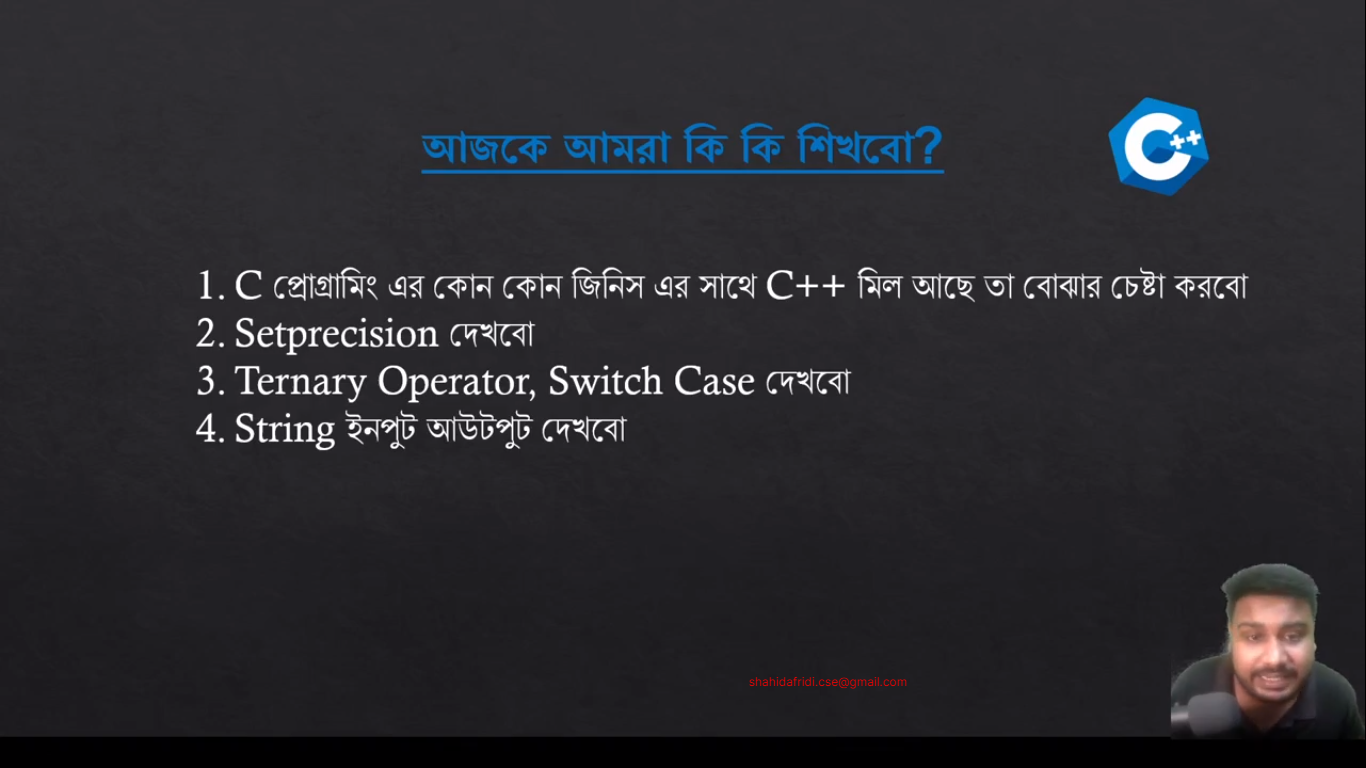
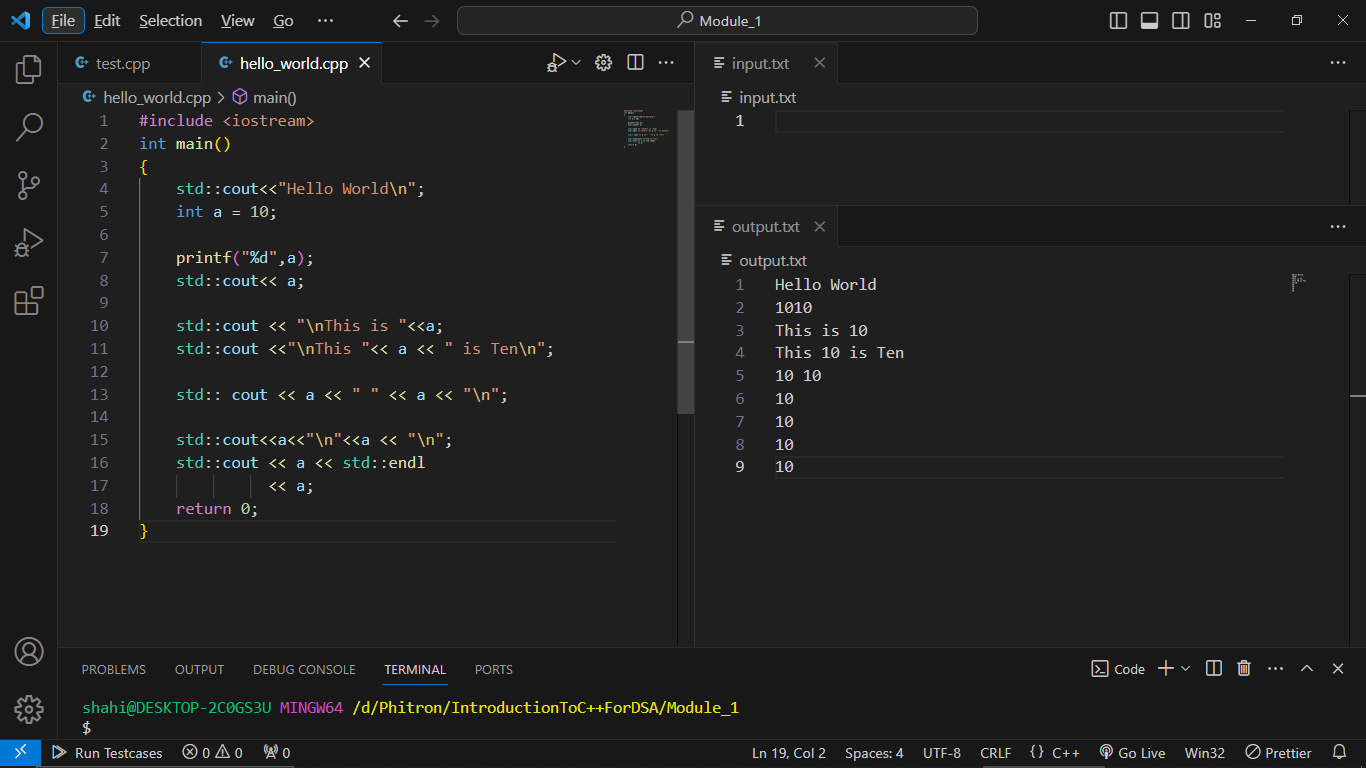
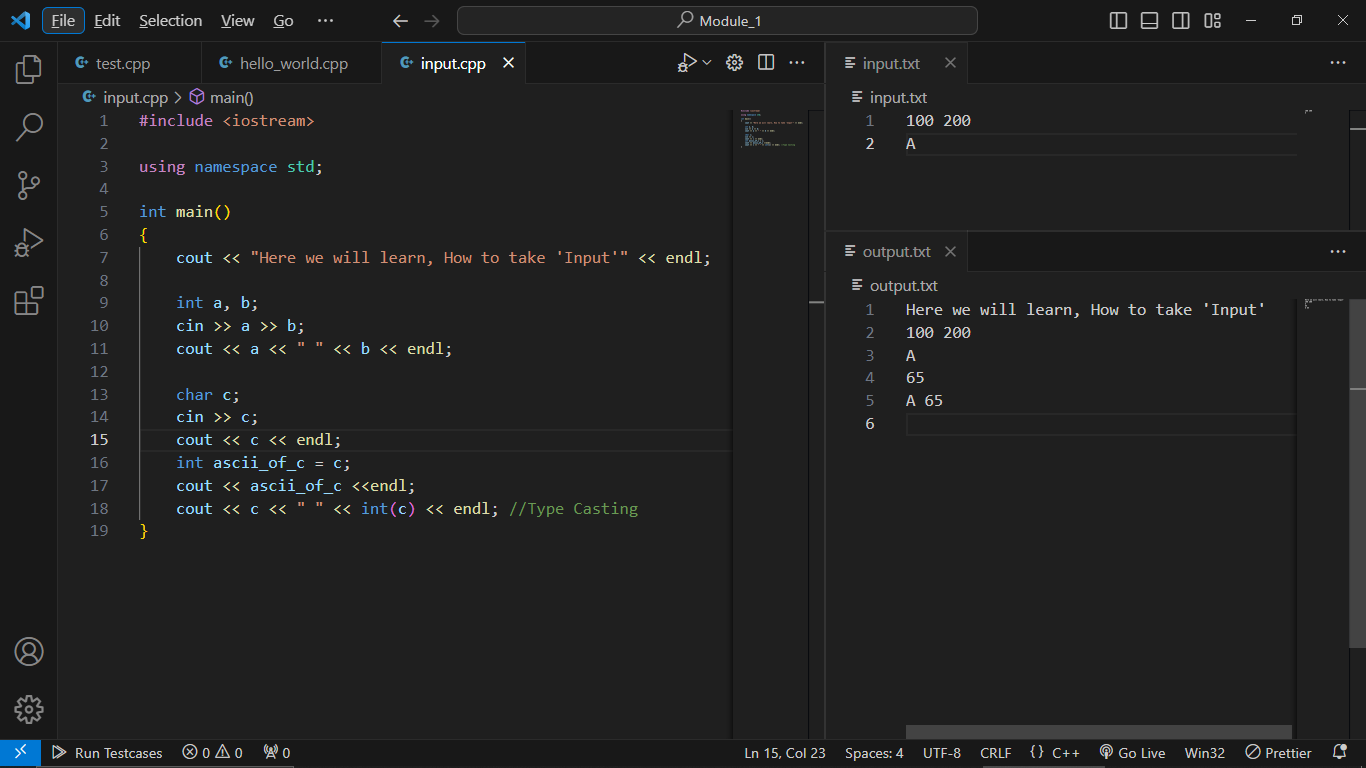
**Introduction To C++ For DSA**

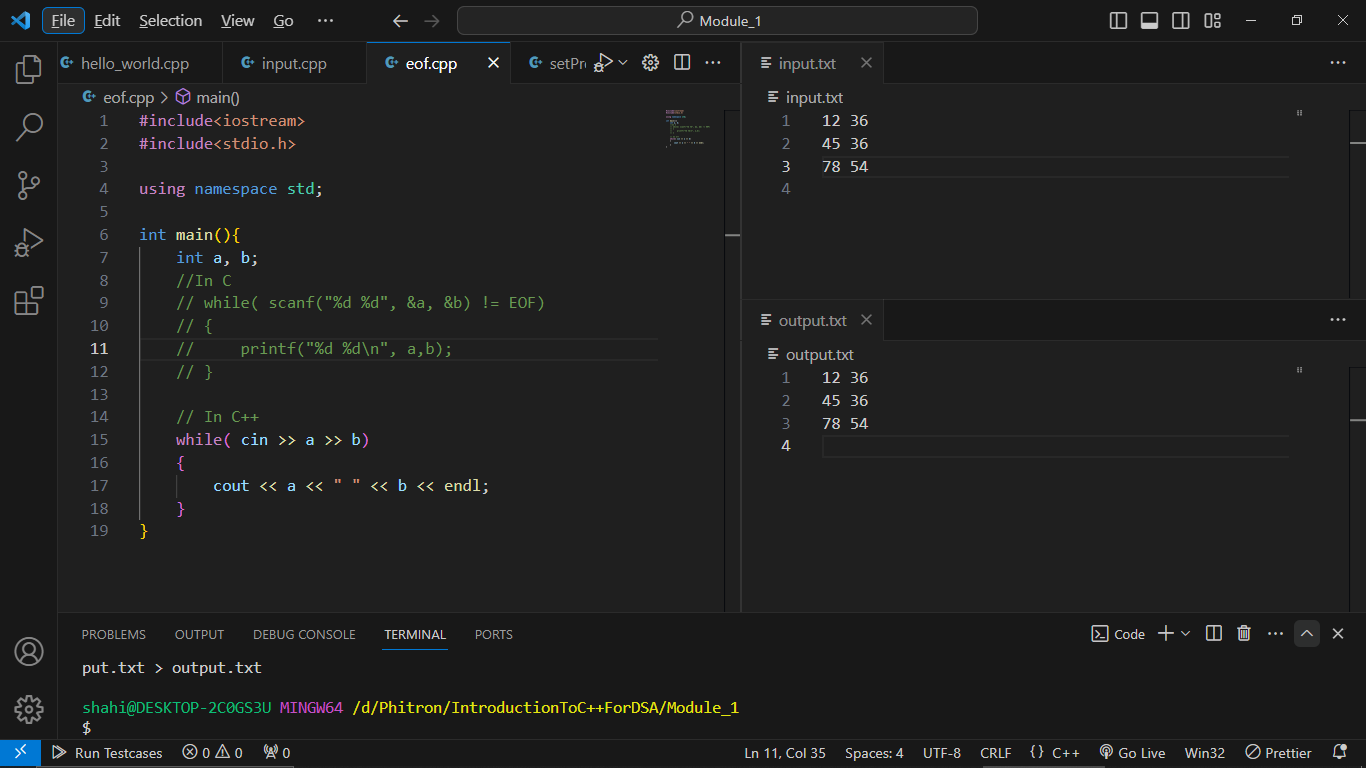
Module 1:



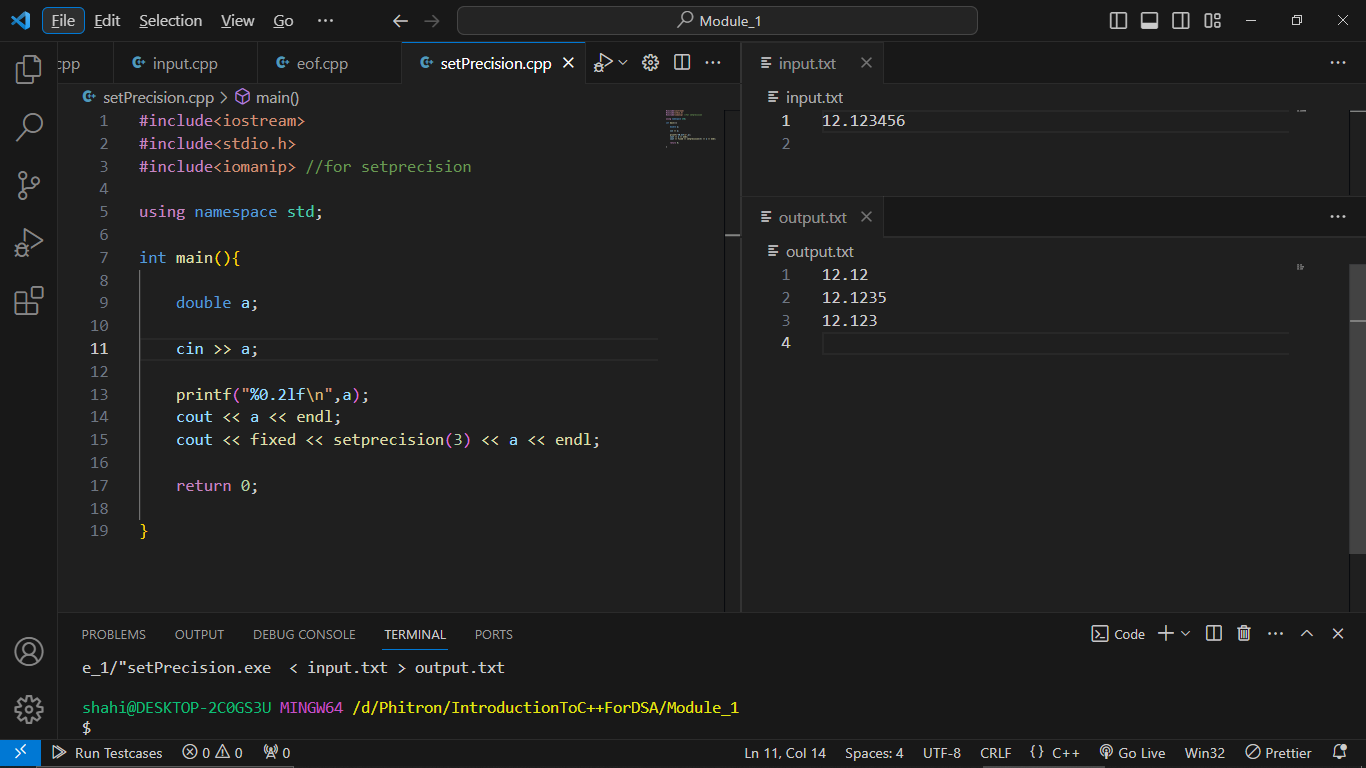




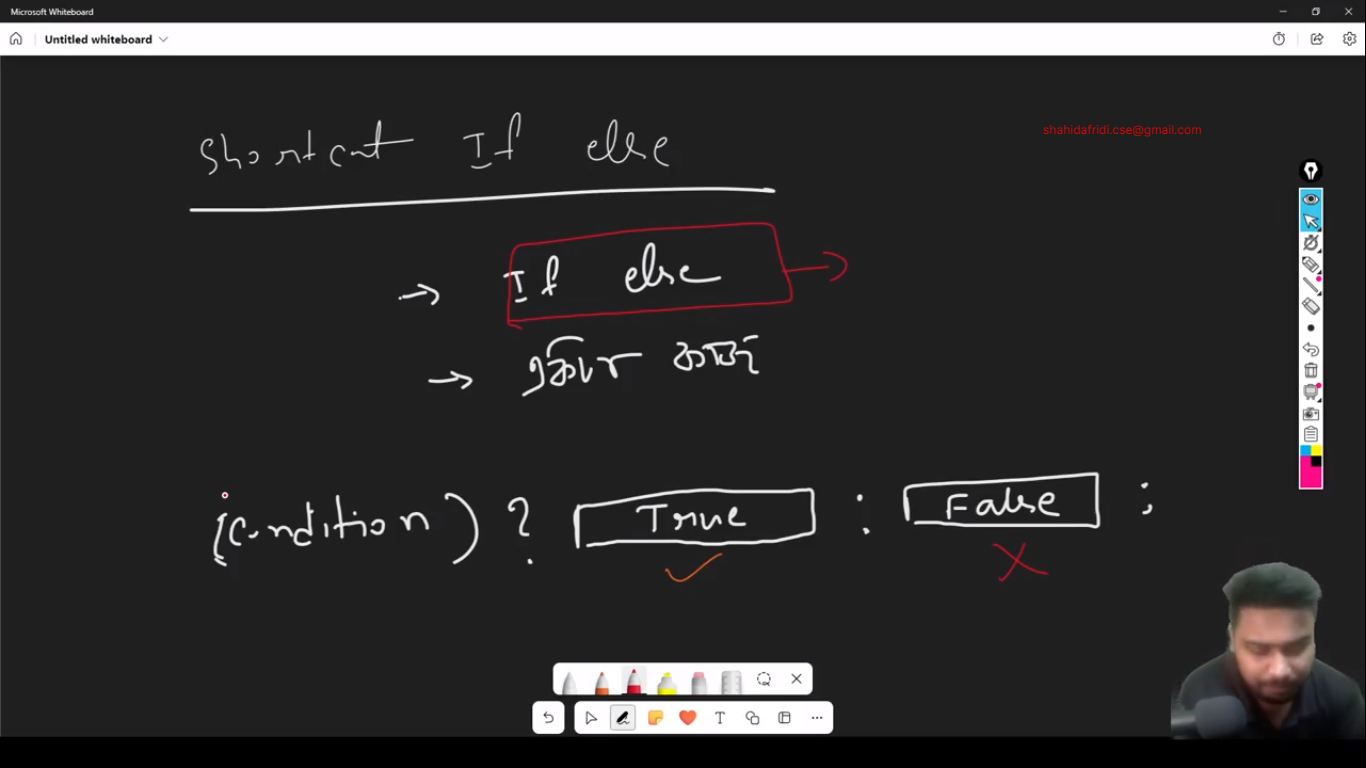
EOF:

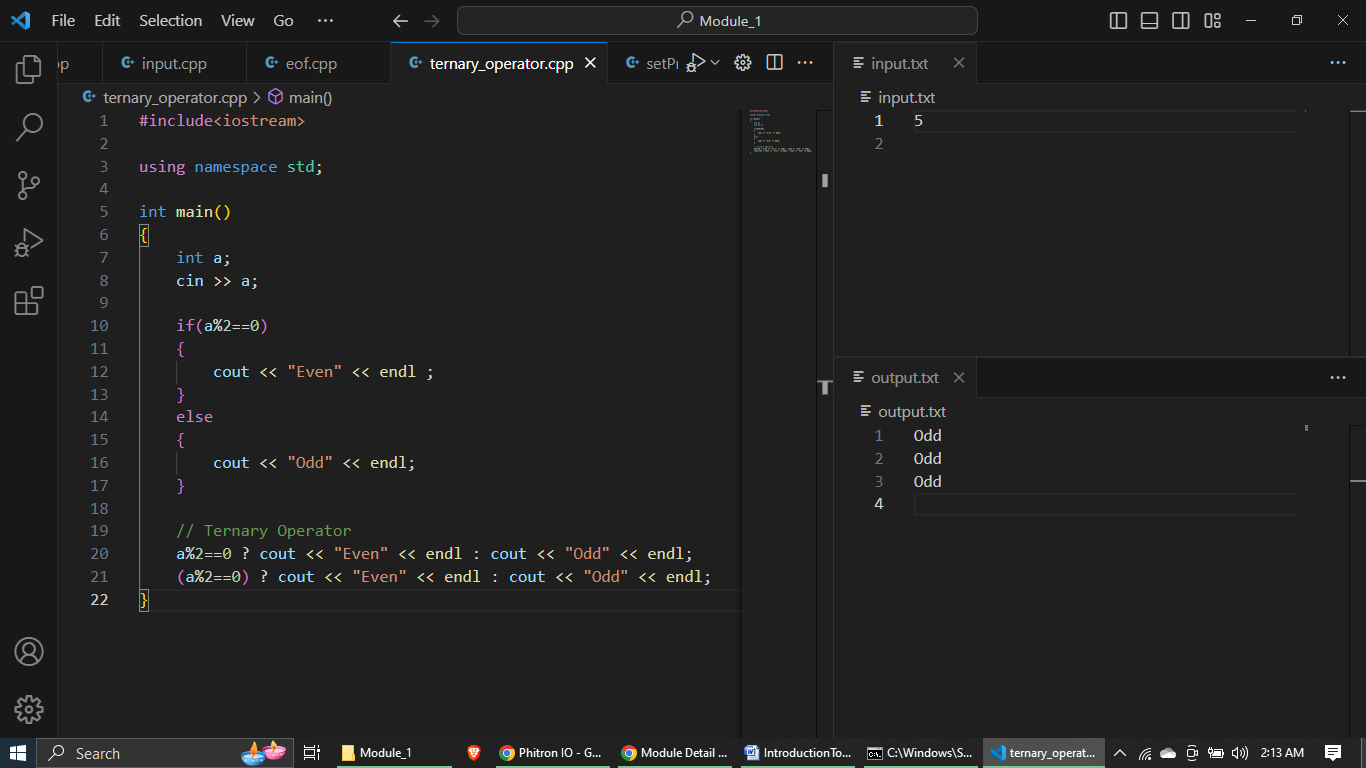


setprecision

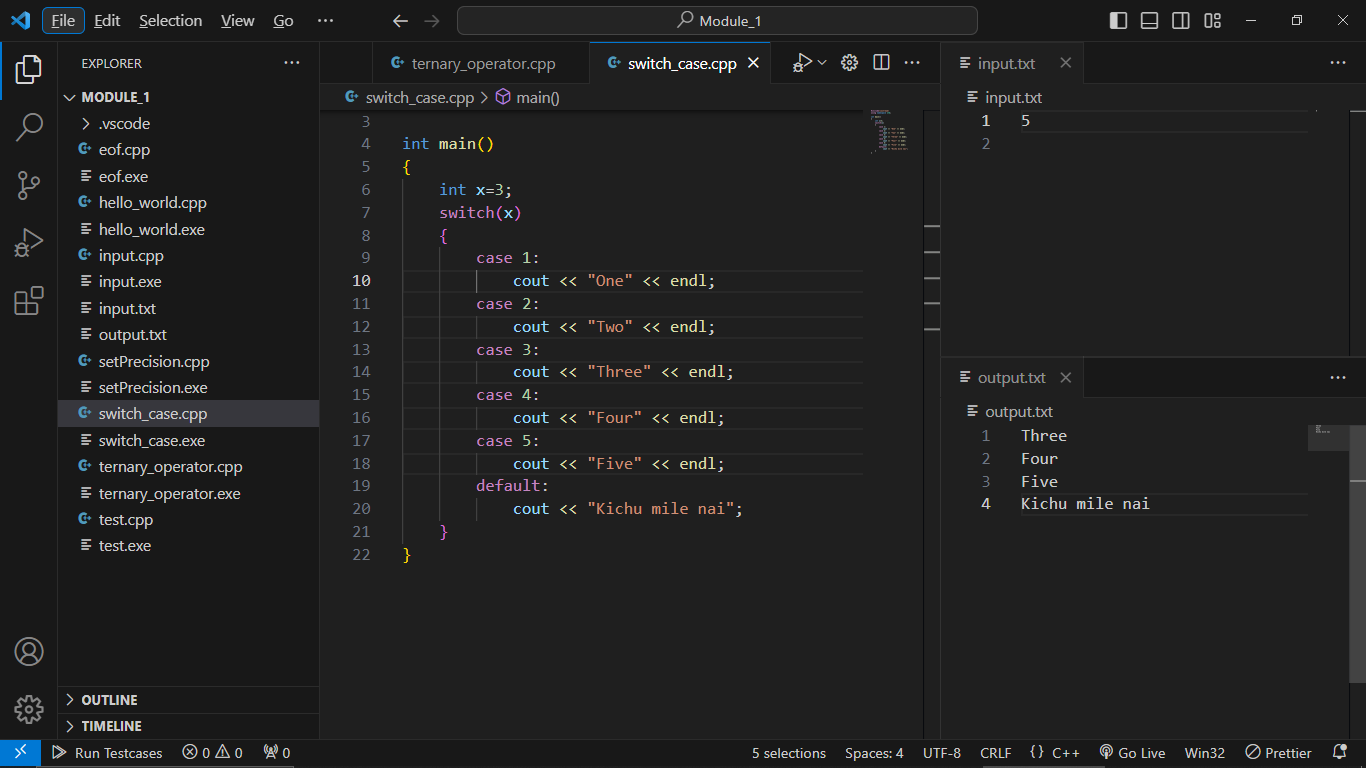


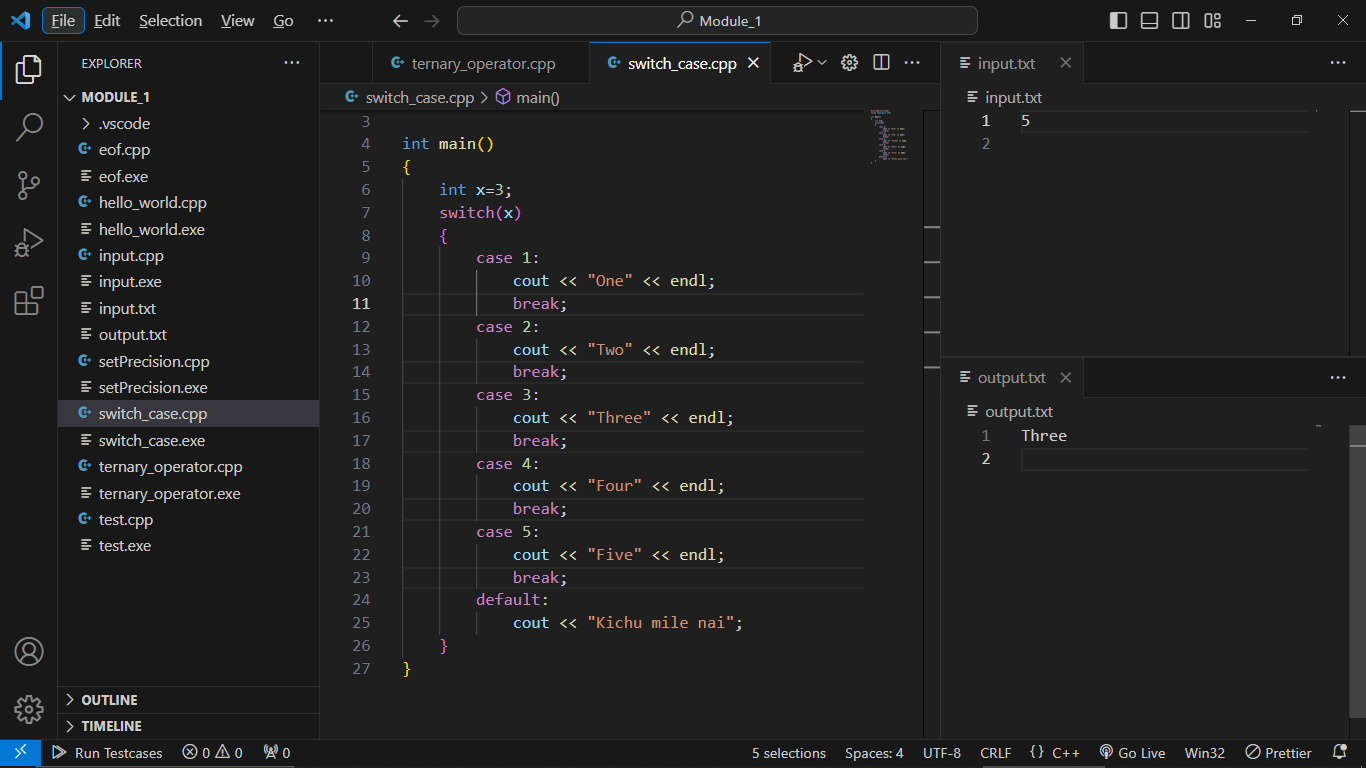
Ternary Operator:

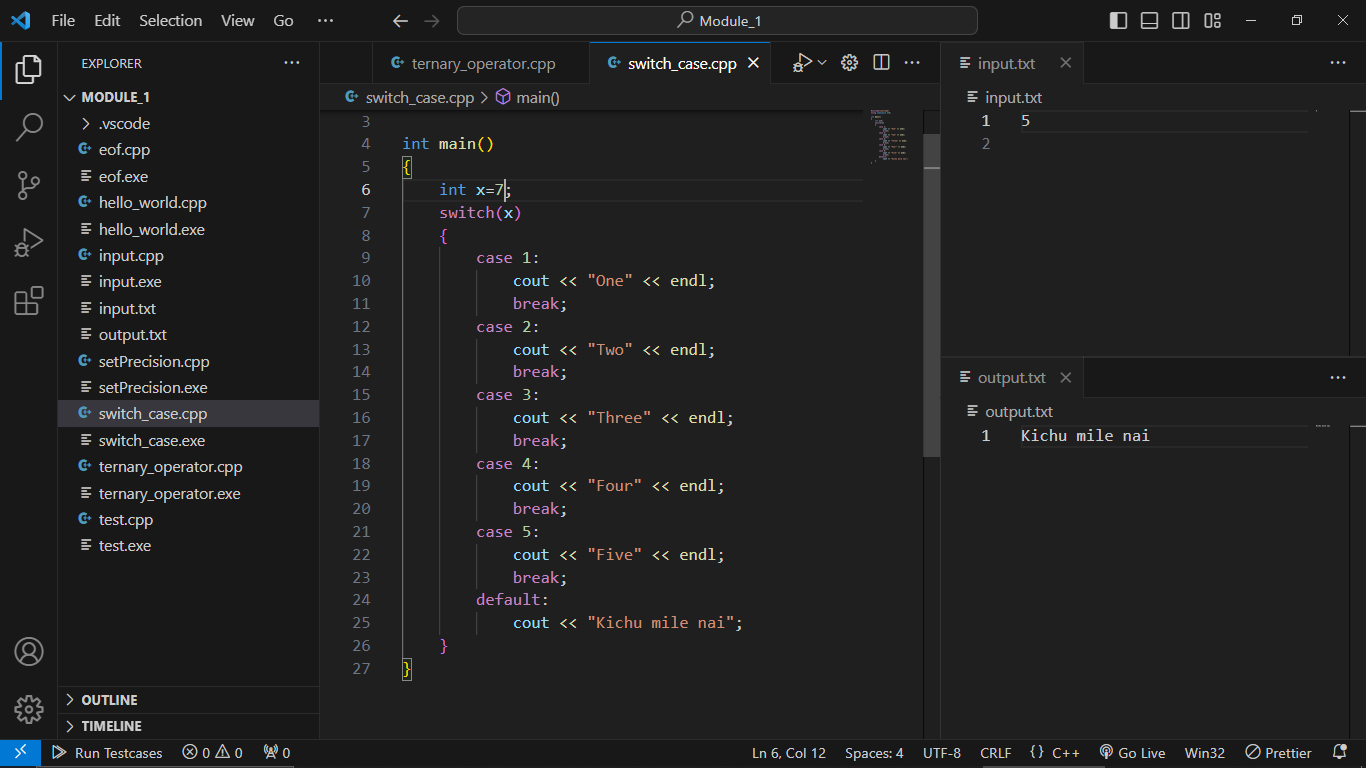




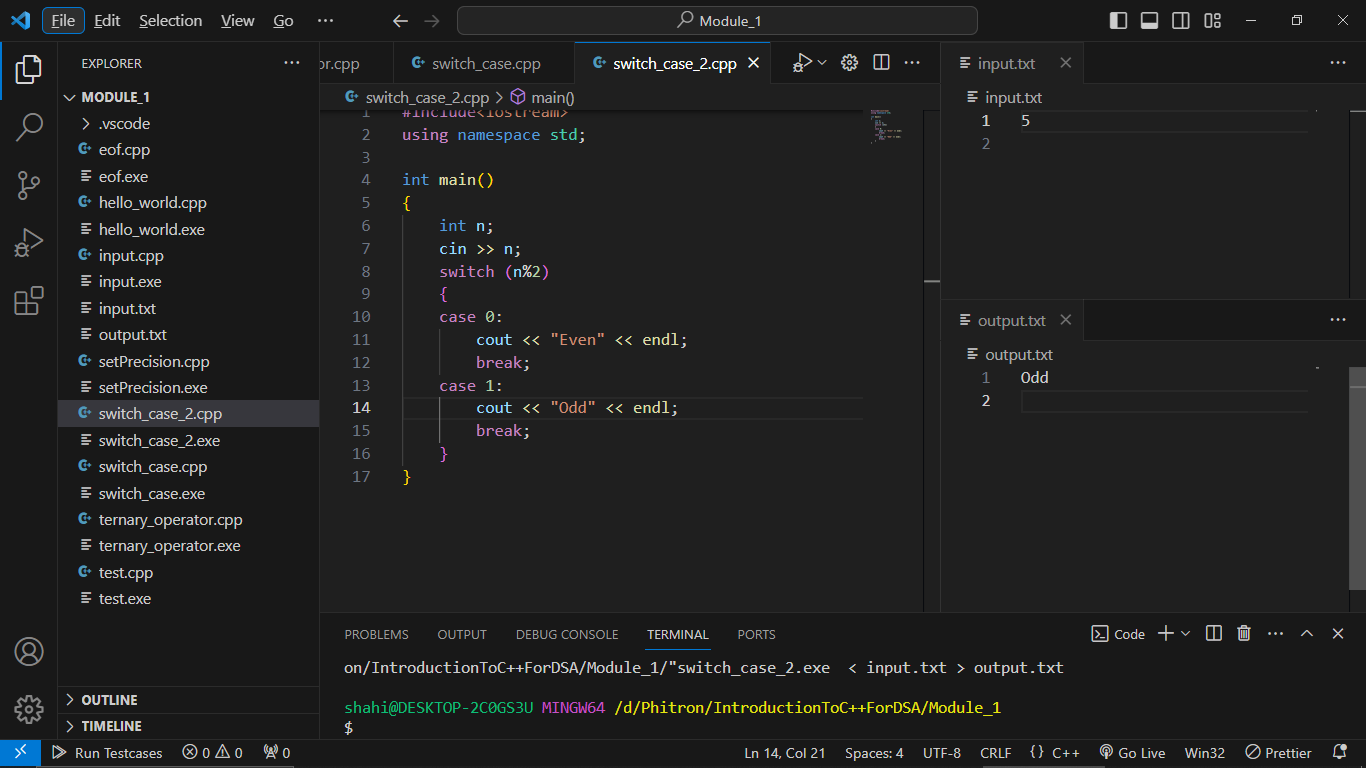
Switch Case:



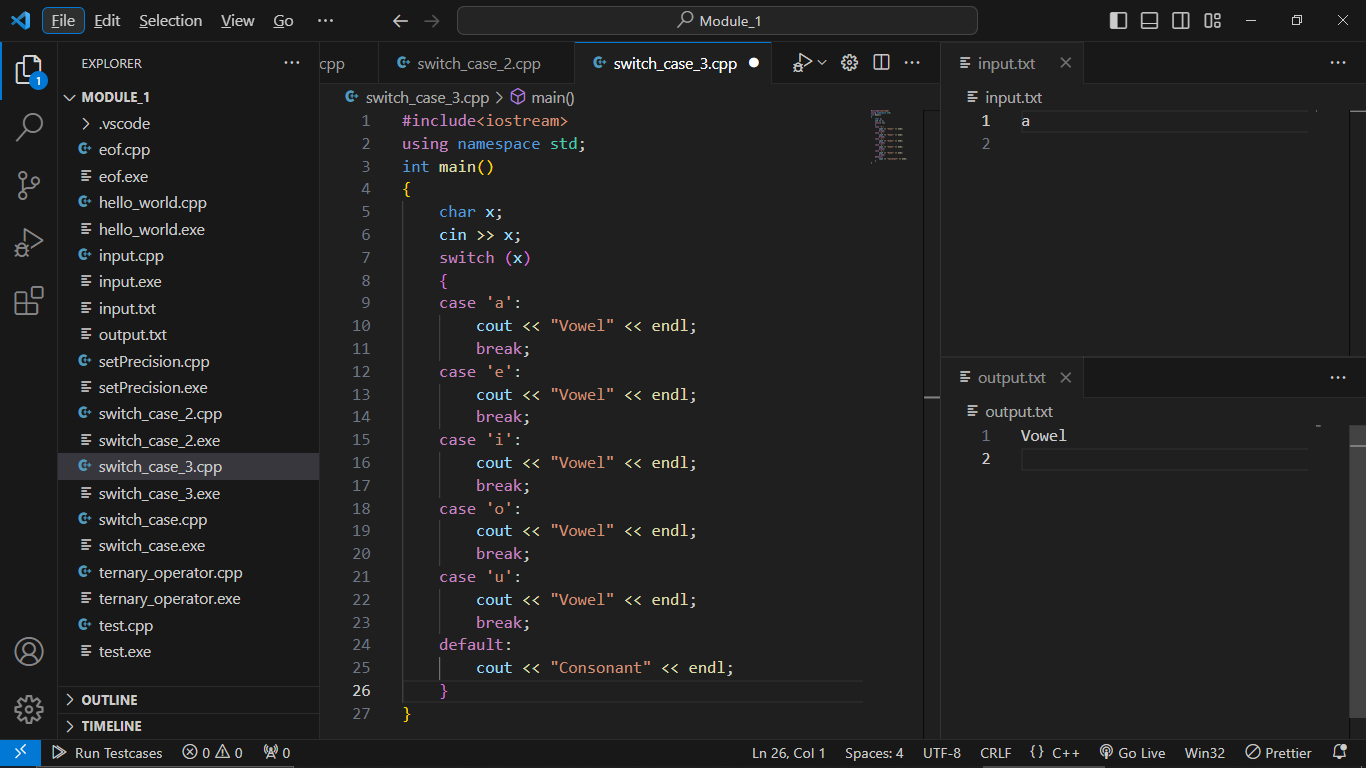




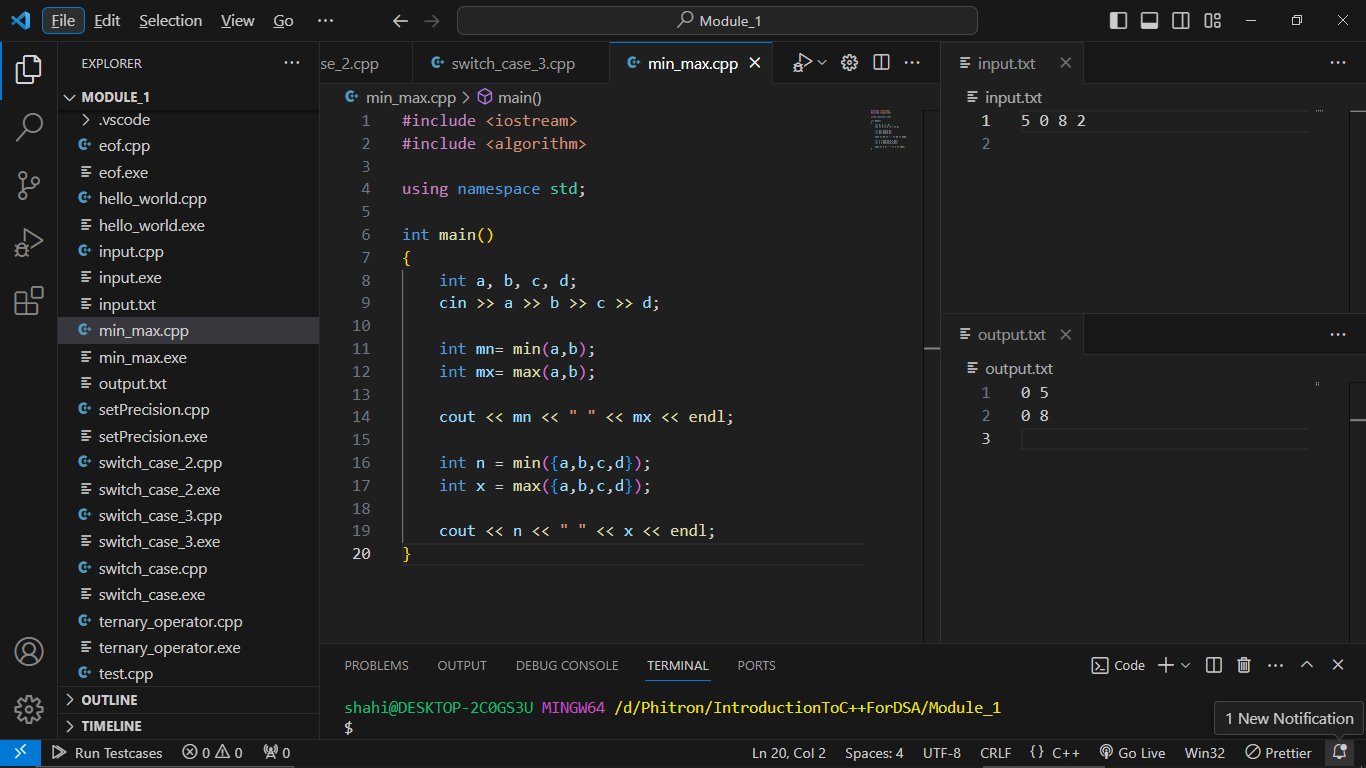
Even Odd Using Switch Case:



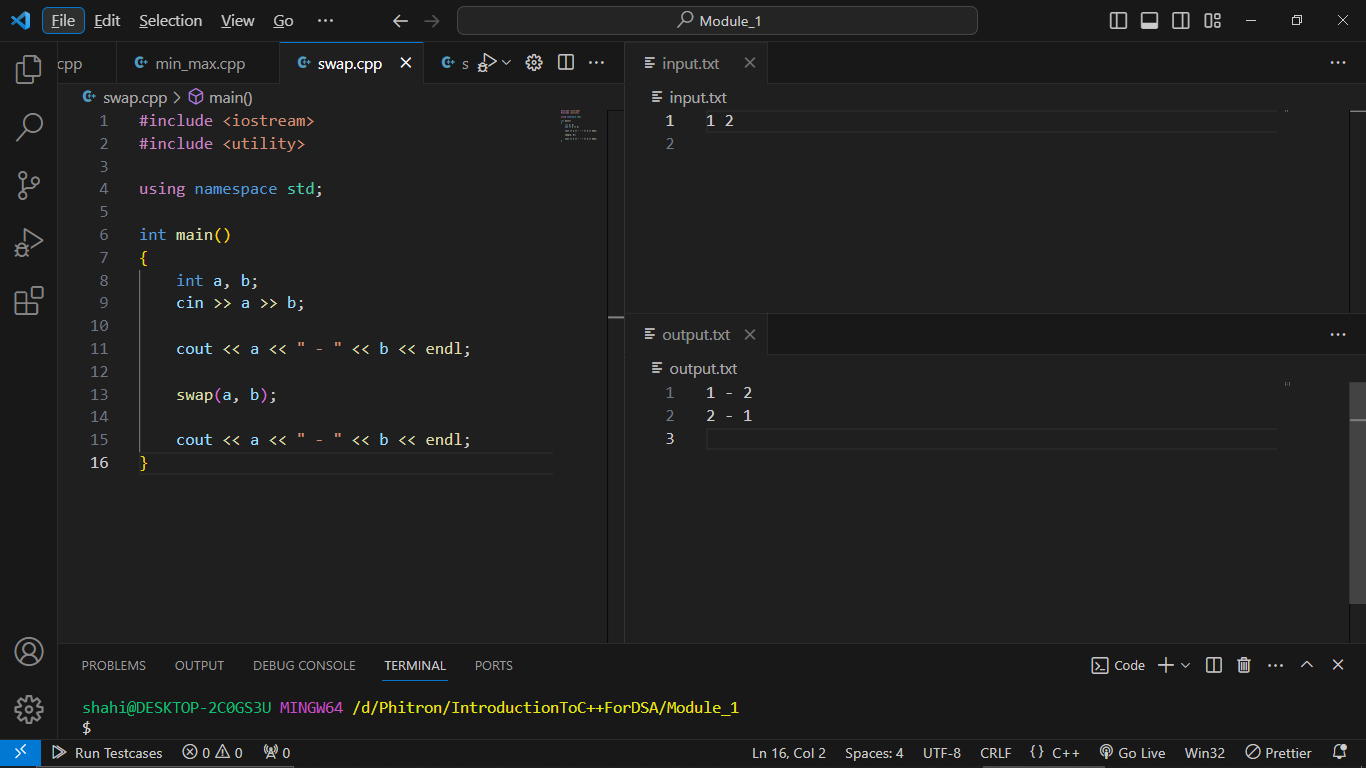
Vowel Consonant Using Switch Case:



Min Max Using Build in Function:



Swap Using Function :



Bits Header File:

#include<bits/stdc++.h>

using namespace std;

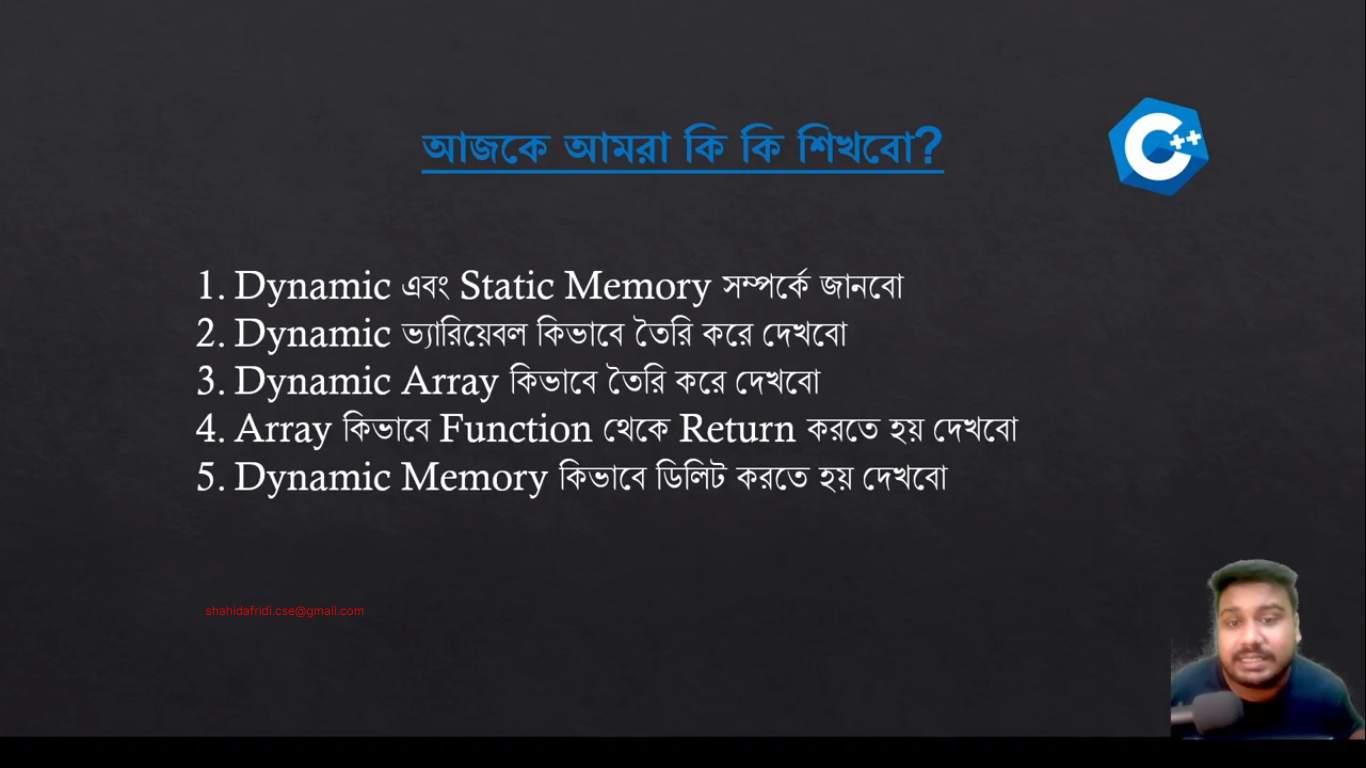
int main()

{

    cout << "Hello”;

}

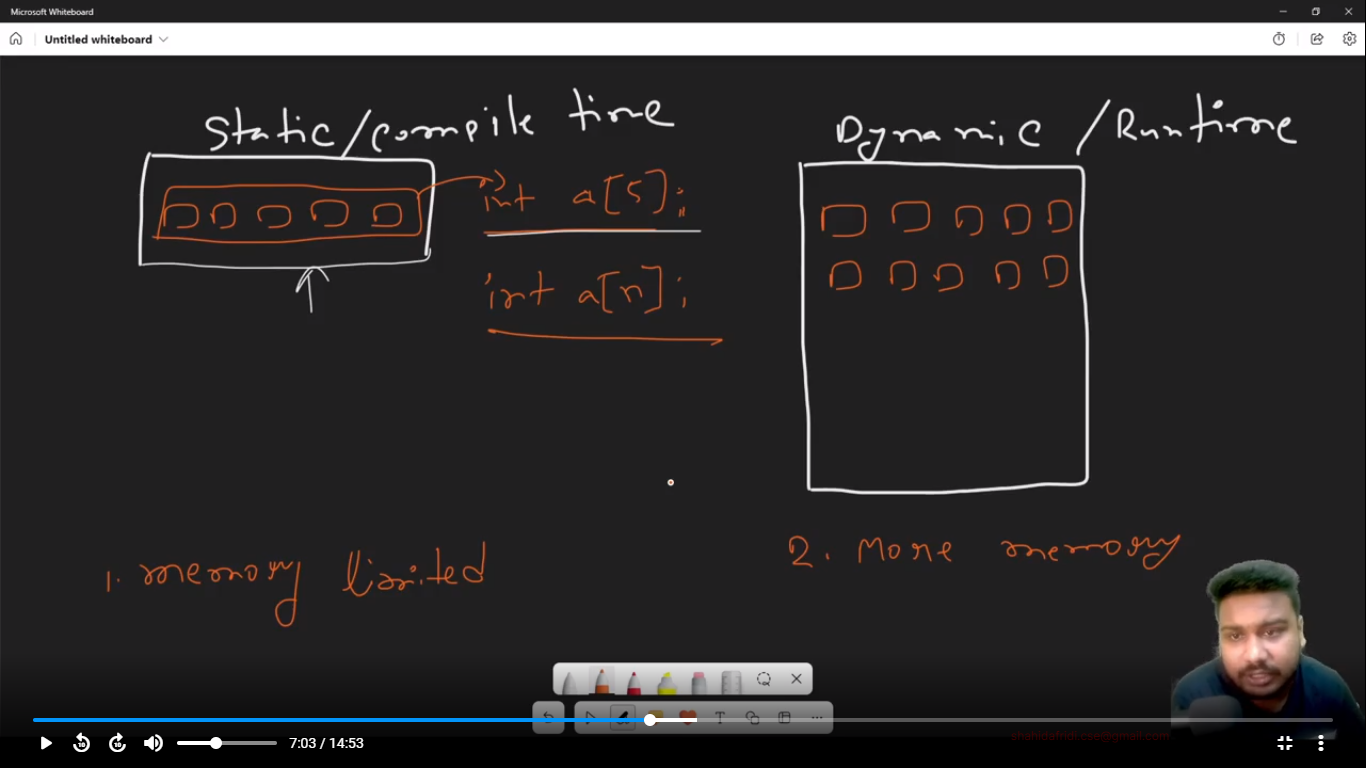
**Module 2:**

****

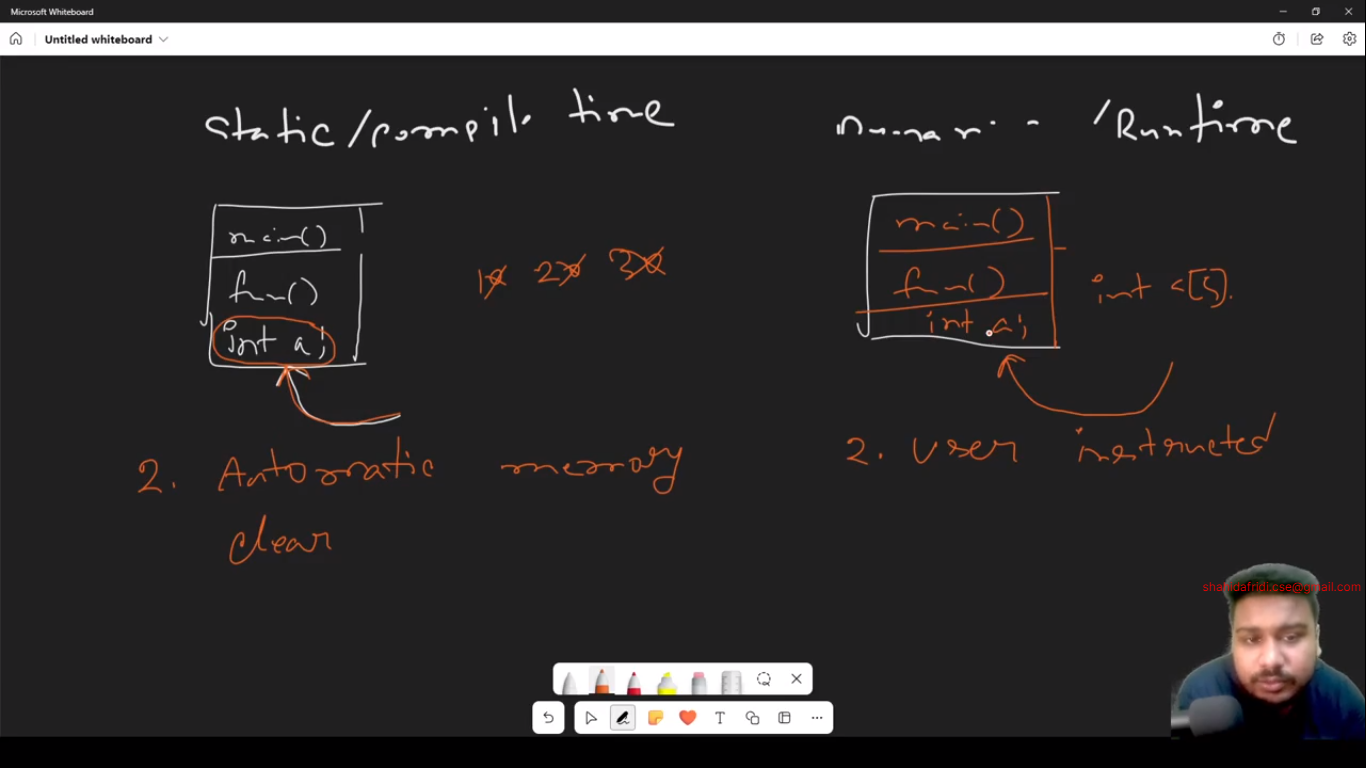
Stack -> Static Memory

Hear -> Dynamic Memory





* Dynamic Memory can increase the size of Array.
* Static Memory can’t increase the size of Array.



What is Static Memory Allocation?

When the allocation of memory performs at the compile time, then it is known as static memory. In this, the memory is allocated for variables by the compiler.

**What is Dynamic Memory Allocation?**

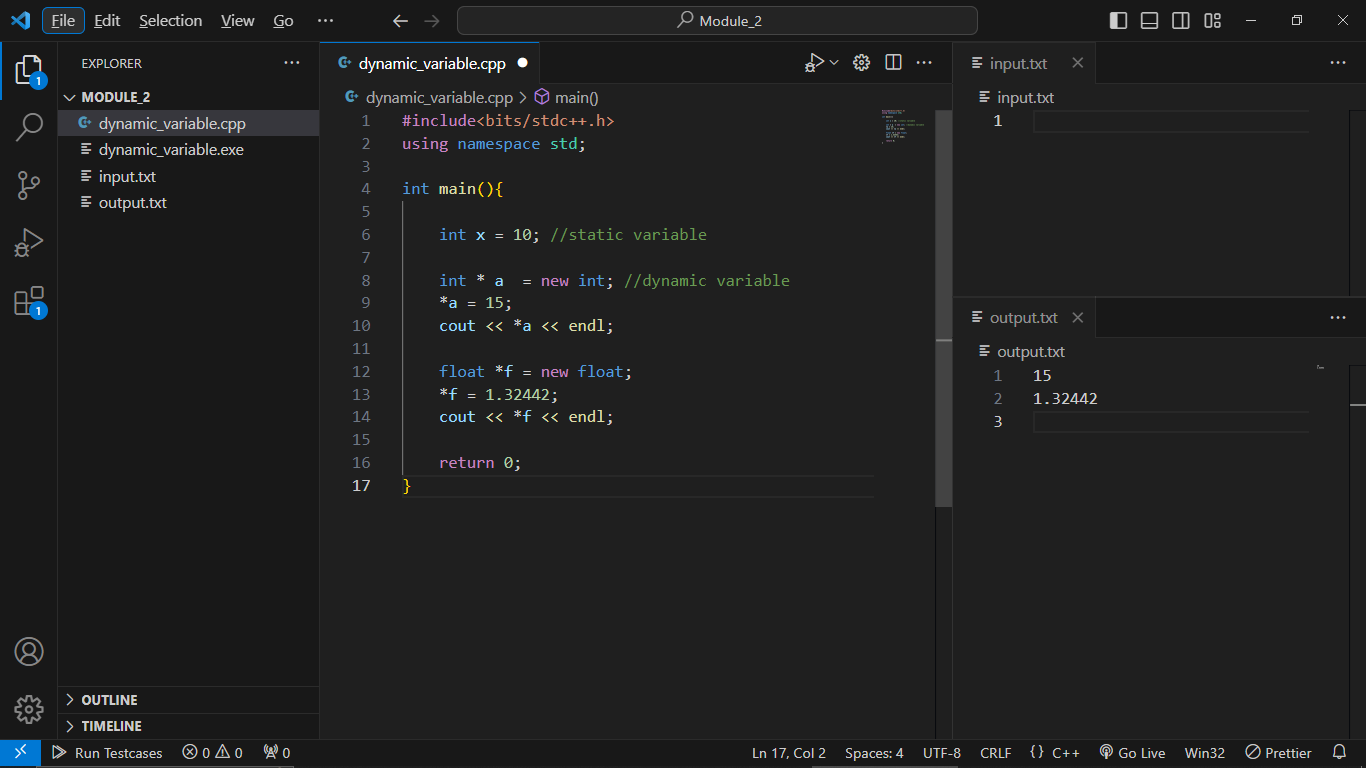
When the memory allocation is done at the execution or run time, then it is called dynamic memory allocation.

**Difference between Static and Dynamic Memory Allocation**

|  |  |  |
| --- | --- | --- |
|  | **Static Memory Allocation** | **Dynamic Memory Allocation** |
| 1 | When the allocation of memory performs at the compile time, then it is known as static memory. | When the memory allocation is done at the execution or run time, then it is called dynamic memory allocation. |
| 2 | The memory is allocated at the compile time. | The memory is allocated at the runtime. |
| 3 | In static memory allocation, while executing a program, the memory cannot be changed. | In dynamic memory allocation, while executing a program, the memory can be changed. |
| 4 | Static memory allocation is preferred in an array. | Dynamic memory allocation is preferred in the linked list. |
| 5 | It saves running time as it is fast. | It is slower than static memory allocation. |
| 6 | Static memory allocation allots memory from the stack. | Dynamic memory allocation allots memory from the heap. |
| 7 | Once the memory is allotted, it will remain from the beginning to end of the program. | Here, the memory can be allotted at any time in the program. |
| 8 | Static memory allocation is less efficient as compared to Dynamic memory allocation. | Dynamic memory allocation is more efficient as compared to the Static memory allocation. |
| 9 | This memory allocation is simple. | This memory allocation is complicated. |

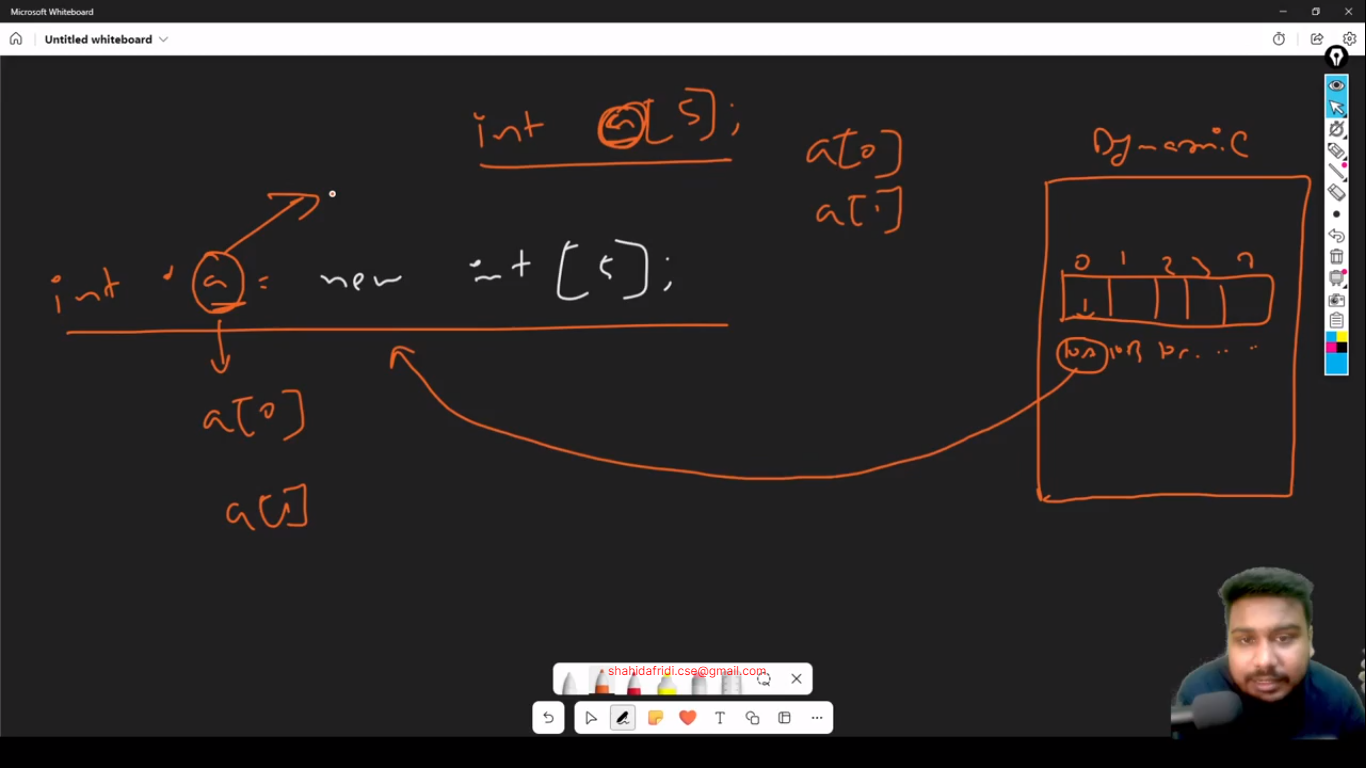
|  |  |  |
| --- | --- | --- |
|  | Static Memory Allocation | Dynamic Memory Allocation |
| 1 | In the static memory allocation, variables get allocated permanently, till the program executes or function call finishes. | In the Dynamic memory allocation, variables get allocated only if your program unit gets active. |
| 2 | Static Memory Allocation is done before program execution. | Dynamic Memory Allocation is done during program execution. |
| 3 | It uses [stack](https://www.geeksforgeeks.org/stack-data-structure/) for managing the static allocation of memory | It uses [heap](https://www.geeksforgeeks.org/heap-data-structure/) for managing the dynamic allocation of memory |
| 4 | It is less efficient | It is more efficient |
| 5 | In Static Memory Allocation, there is no memory re-usability | In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required |
| 6 | In static memory allocation, once the memory is allocated, the memory size cannot change. | In dynamic memory allocation, when memory is allocated the memory size can be changed. |
| 7 | In this memory allocation scheme, we cannot reuse the unused memory. | This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it. |
| 8 | In this memory allocation scheme, execution is faster than dynamic memory allocation. | In this memory allocation scheme, execution is slower than static memory allocation. |
| 9 | In this memory is allocated at compile time. | In this memory is allocated at run time. |
| 10 | In this allocated memory remains from start to end of the program. | In this allocated memory can be released at any time during the program. |
| 11 | **Example:** This static memory allocation is generally used for [array](https://www.geeksforgeeks.org/introduction-to-arrays/). | **Example:** This dynamic memory allocation is generally used for [linked list](https://www.geeksforgeeks.org/data-structures/linked-list/). |

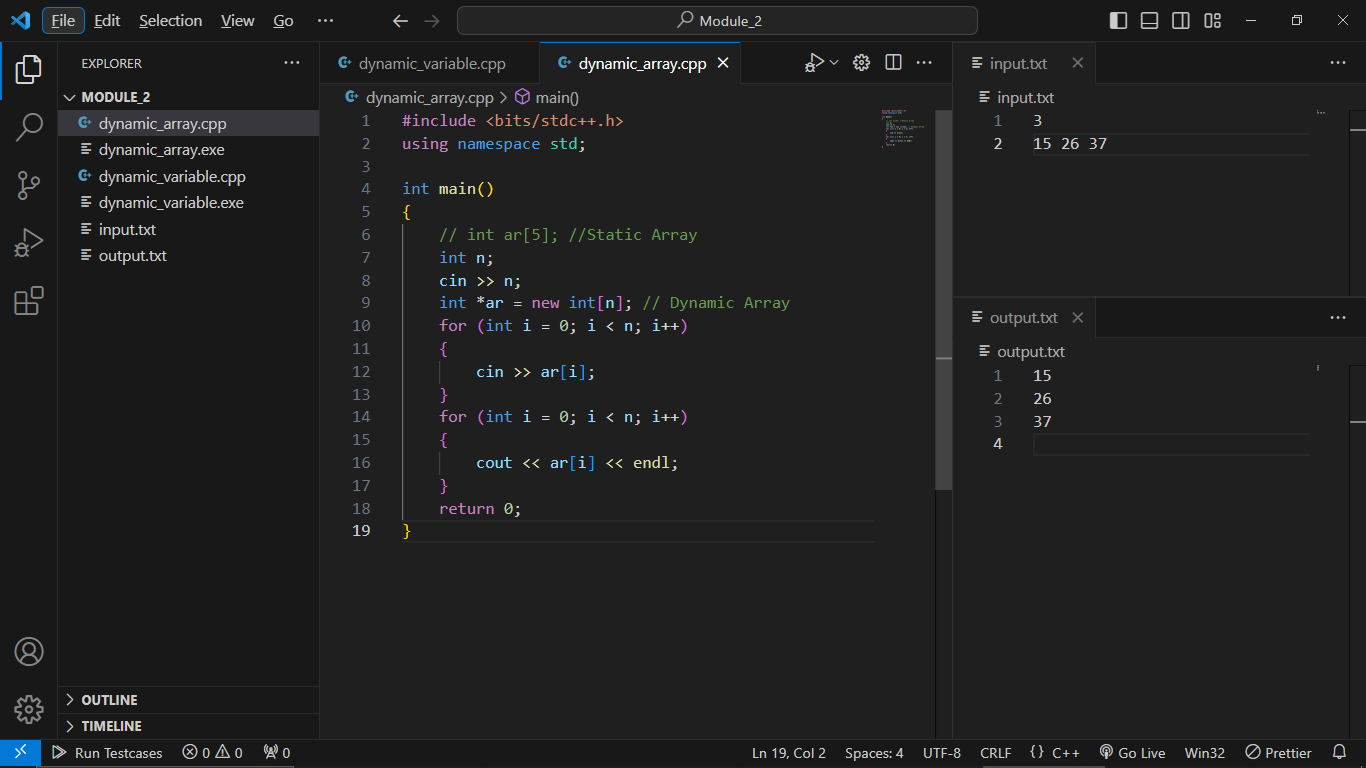
Dynamic Variable:

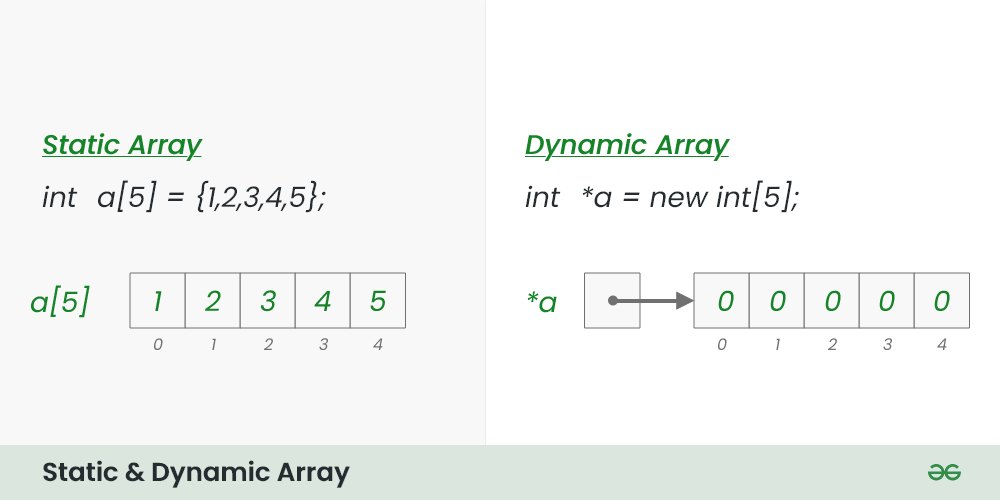


**Module 2-3 Create Dynamic Array and Return Array From Function:**

**Dynamic Array:**

****

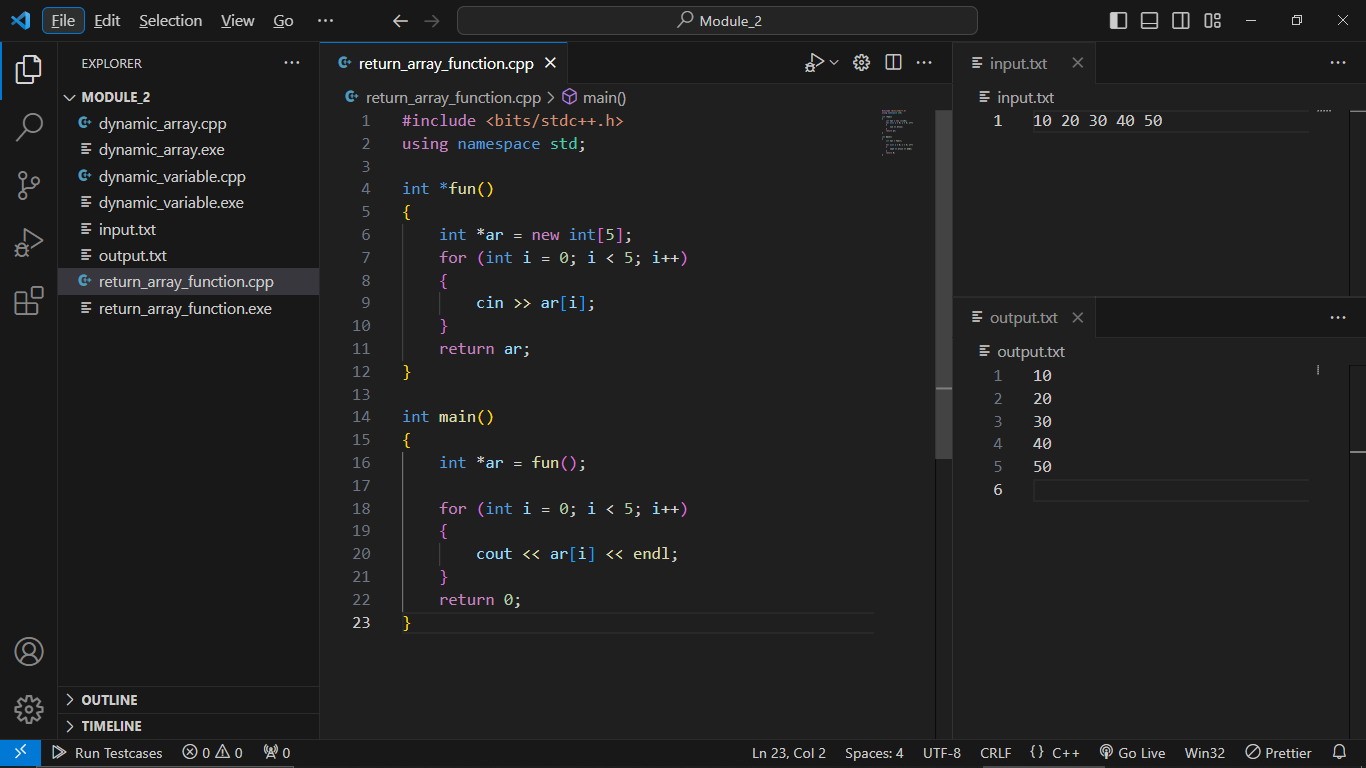
****



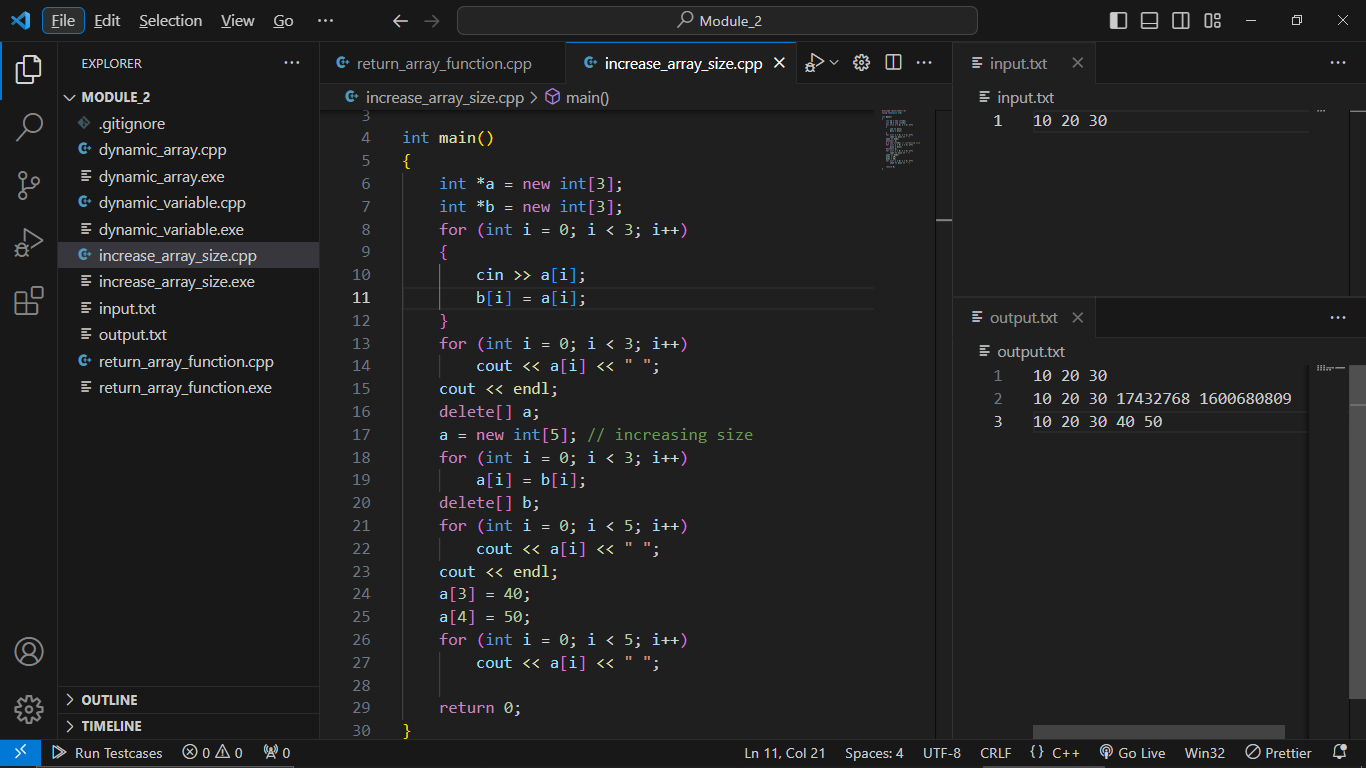
**Key Difference between Static and Dynamic Arrays:**

|  |  |
| --- | --- |
| **Static Array** | **Dynamic Array** |
| 1. The memory allocation occurs during compile time. | 1. The memory allocation occurs during run time. |
| 2. The array size is fixed and cannot be changed. | 2. The array size is not fixed and can be changed. |
| 3. The location is in Stack Memory Space. | 3. The location is in Heap Memory Space. |
| 4. The array elements are set to 0 or to empty strings. | 4. The array elements can be destroyed during erase statement and the memory is then released. |
| 5. This array can be Initialized but not erased. | 5. This array cannot be read or written after destroying. |

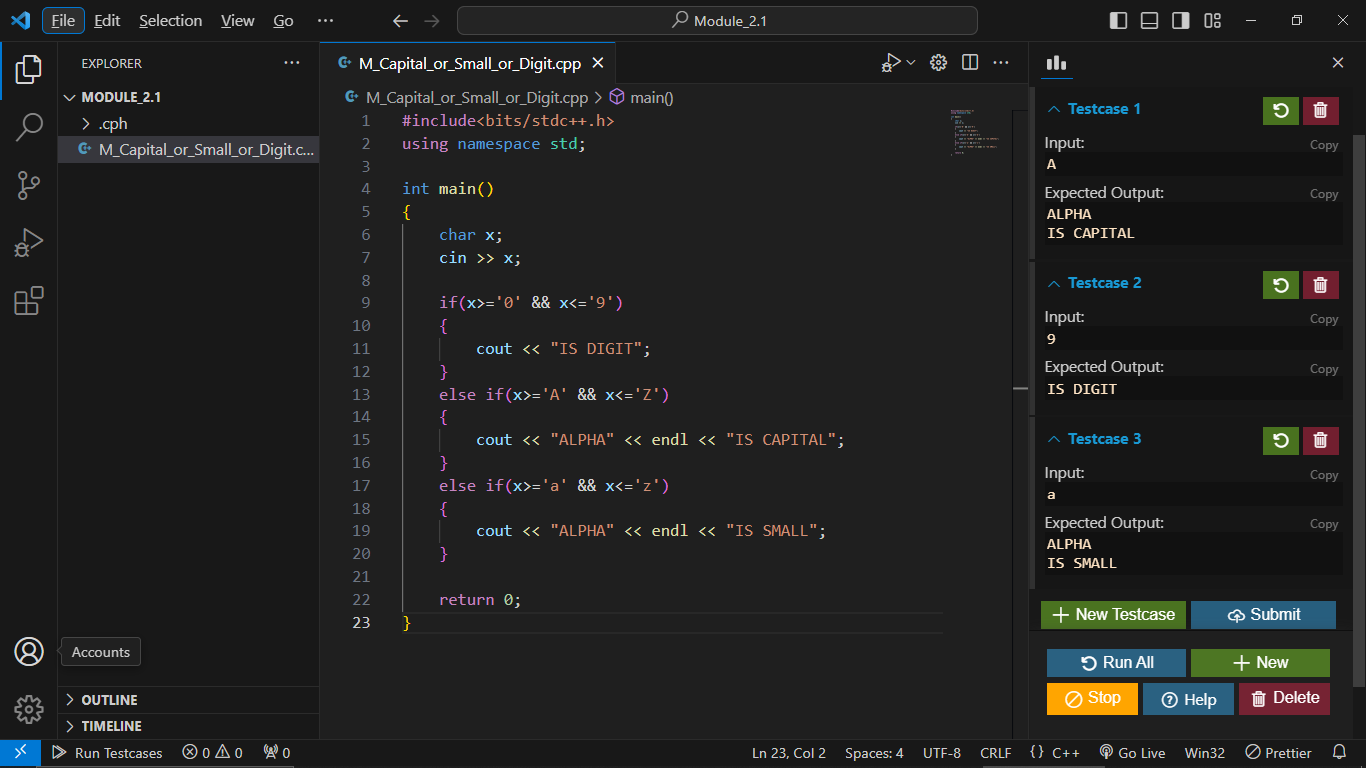
Return Array Function:

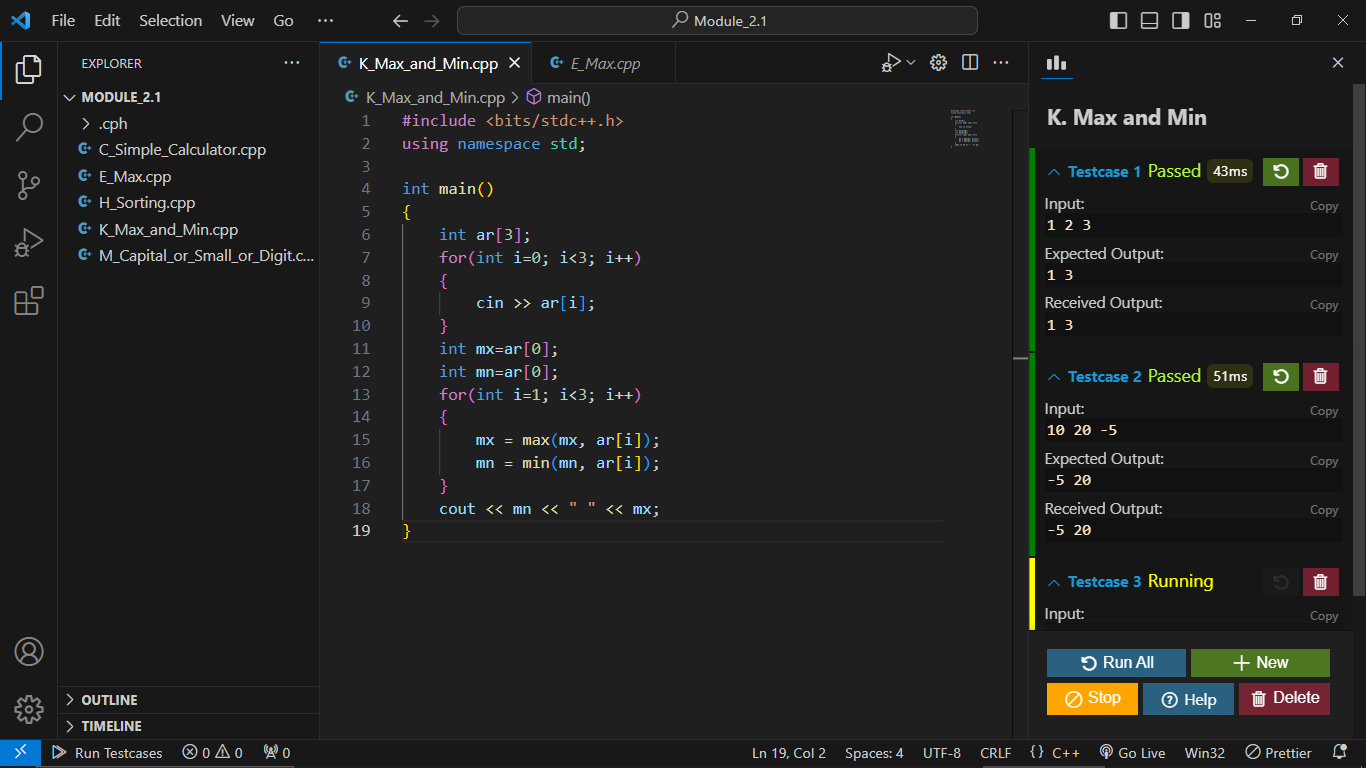


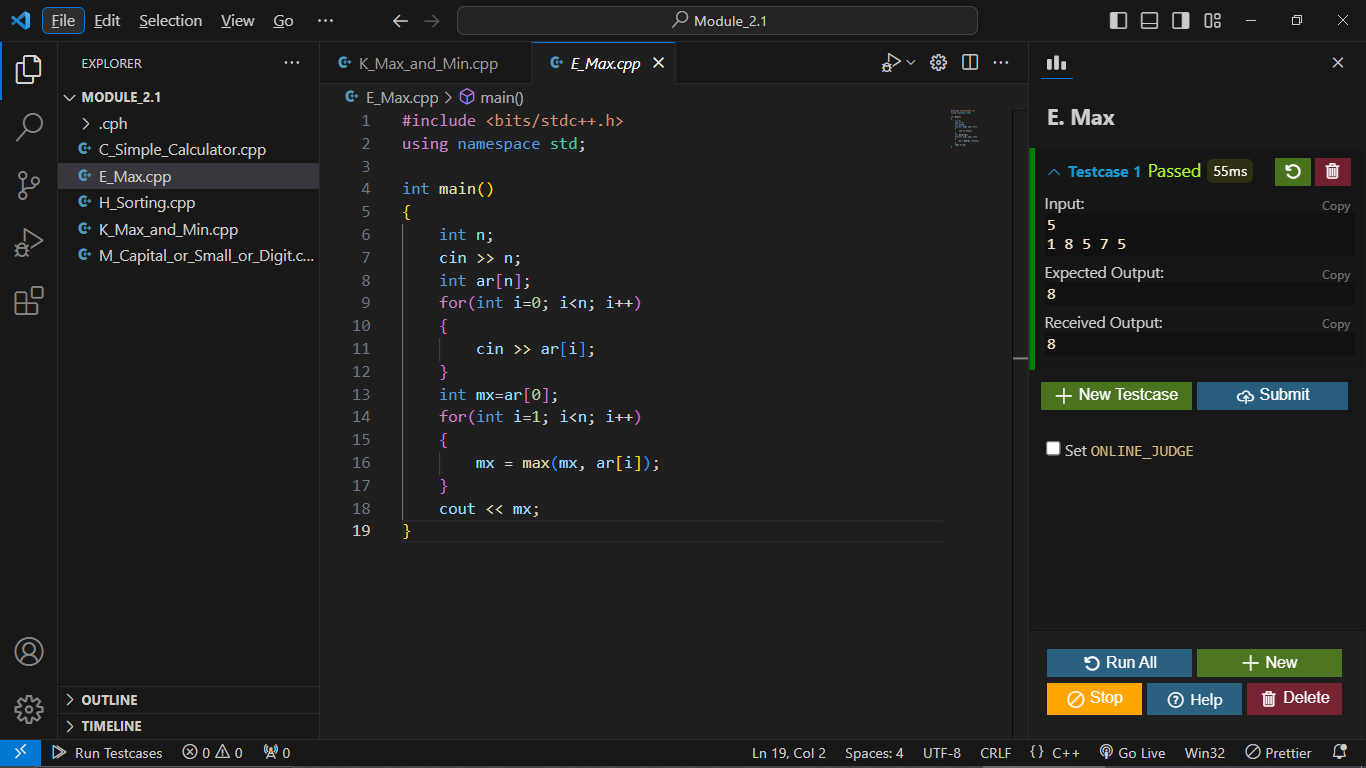
**Module 2-4 Increase Size of Array and Delete Dynamic Memory:**

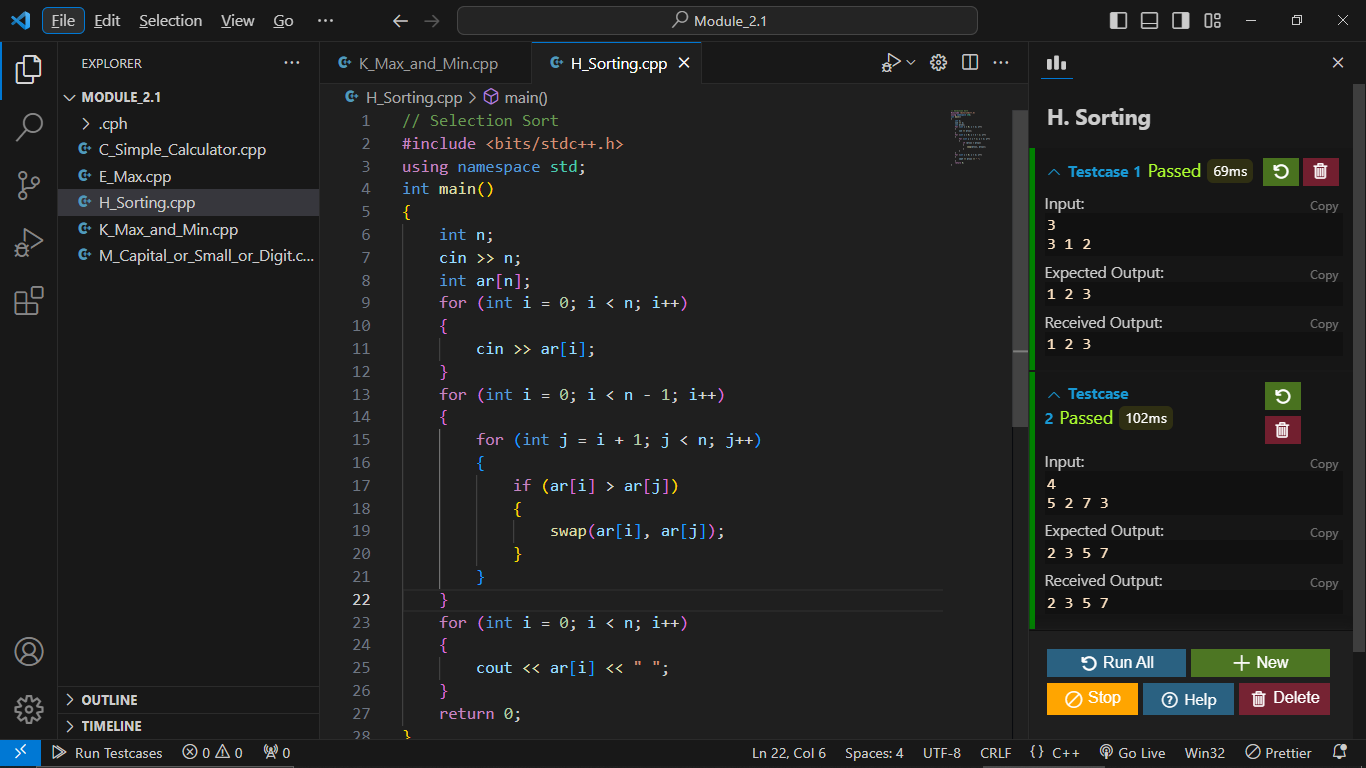
******

**Module 2.5-1 Practice:**





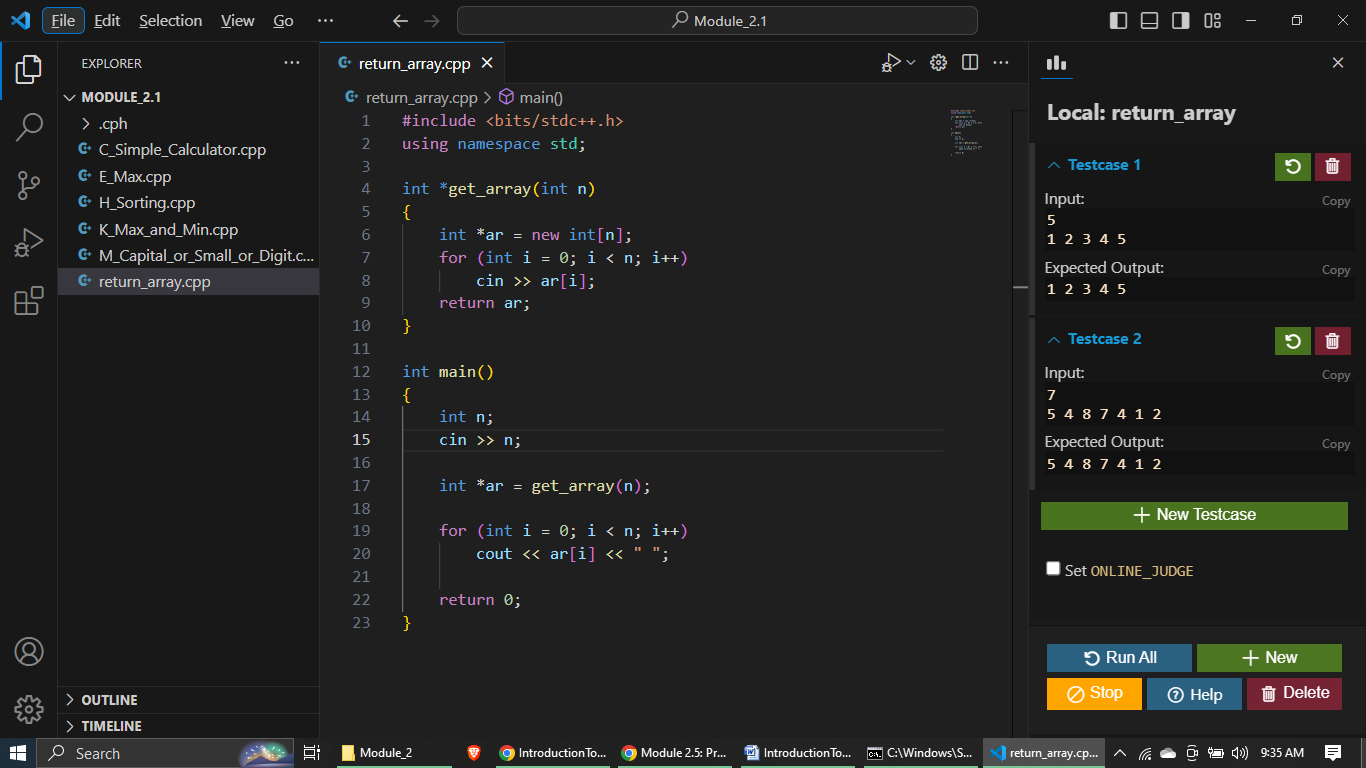




**Practice Problem 1:**

Question: At first in the main function take an integer N as input. Then make a function named get\_array() which will receive that N as a parameter. Then inside the get\_array() function create an integer array of size N. Then the values of that array will be taken as input. After that return that array from that function and receive it in the main function and print the values of the array there.

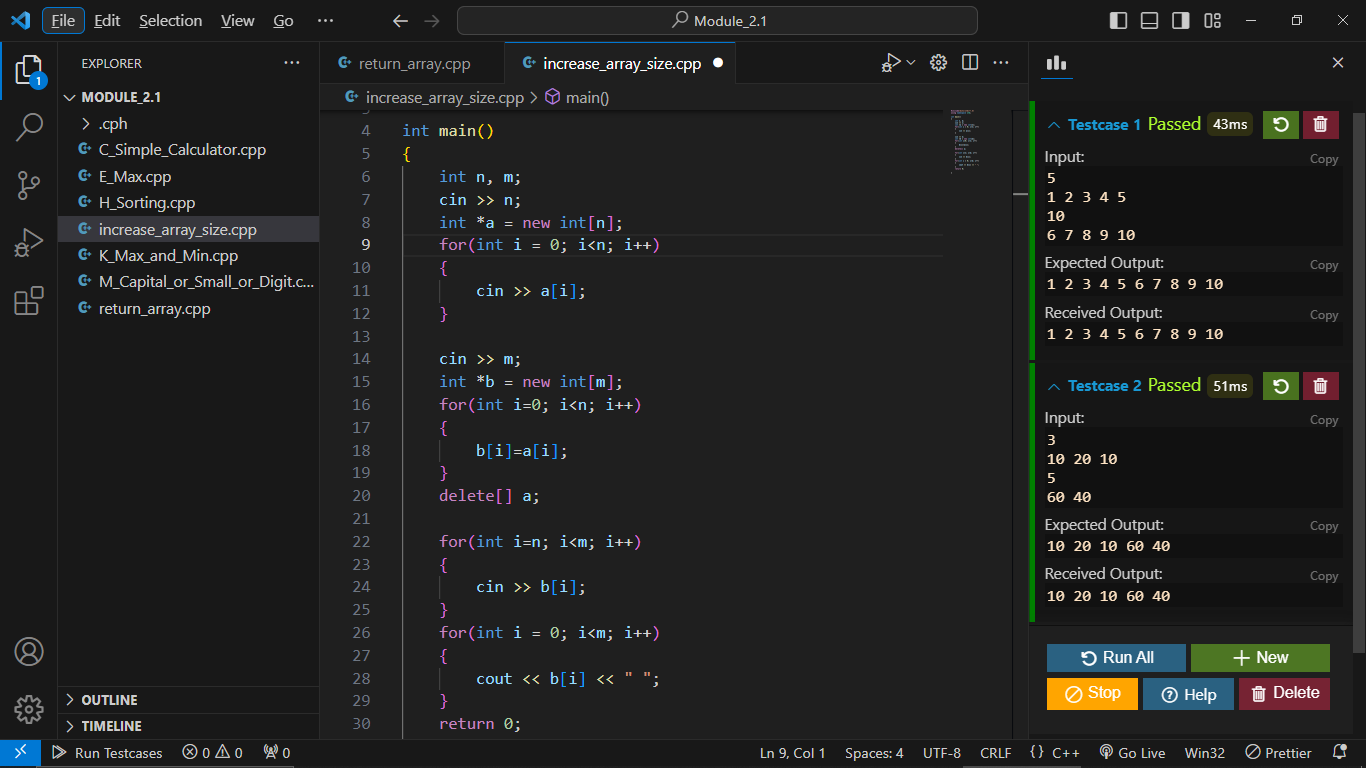
|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **5**  **1 2 3 4 5** | **1 2 3 4 5** |
| **7**  **5 1 2 4 6 9 3** | **5 1 2 4 6 9 3** |



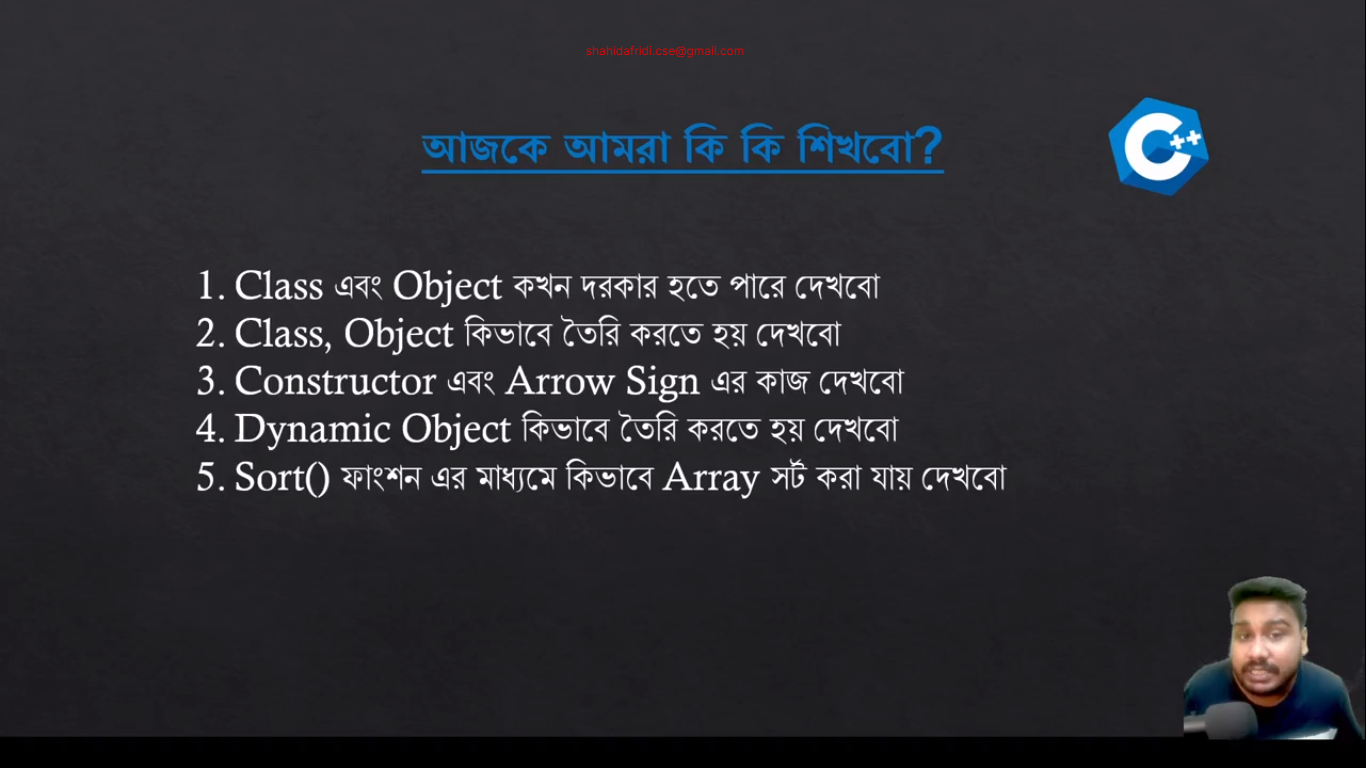
**Practice Problem 2:**

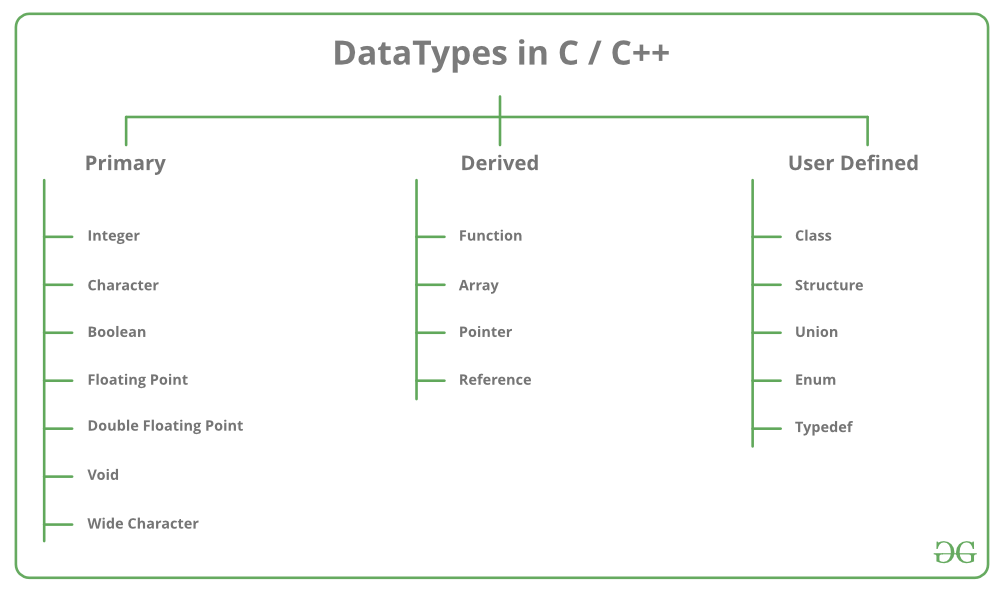
Question: Take an integer array **A** of size **N** as input. Then take an integer **M** as input. You need to take an array **B** of size M and copy all elements of array **A** to array **B**. Delete the array **A** and then take input of the rest of the elements of array **B**. After that print array B.

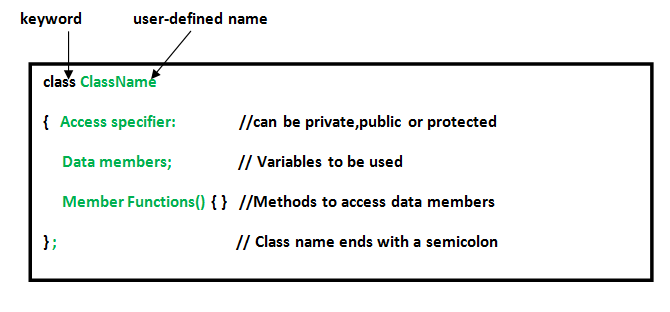
|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5  1 2 3 4 5  10  6 7 8 9 10 | 1 2 3 4 5 6 7 8 9 10 |
| 3  10 20 10  5  60 40 | 10 20 10 60 40 |

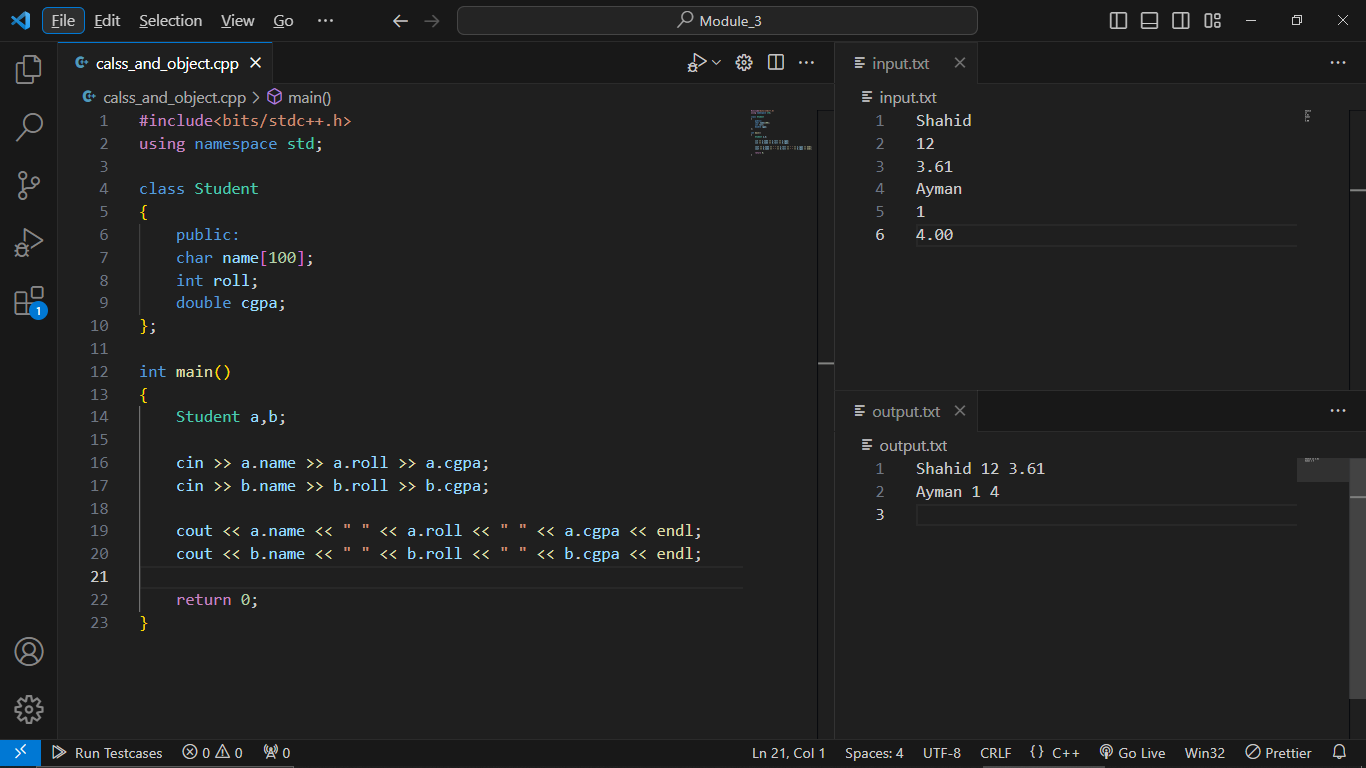


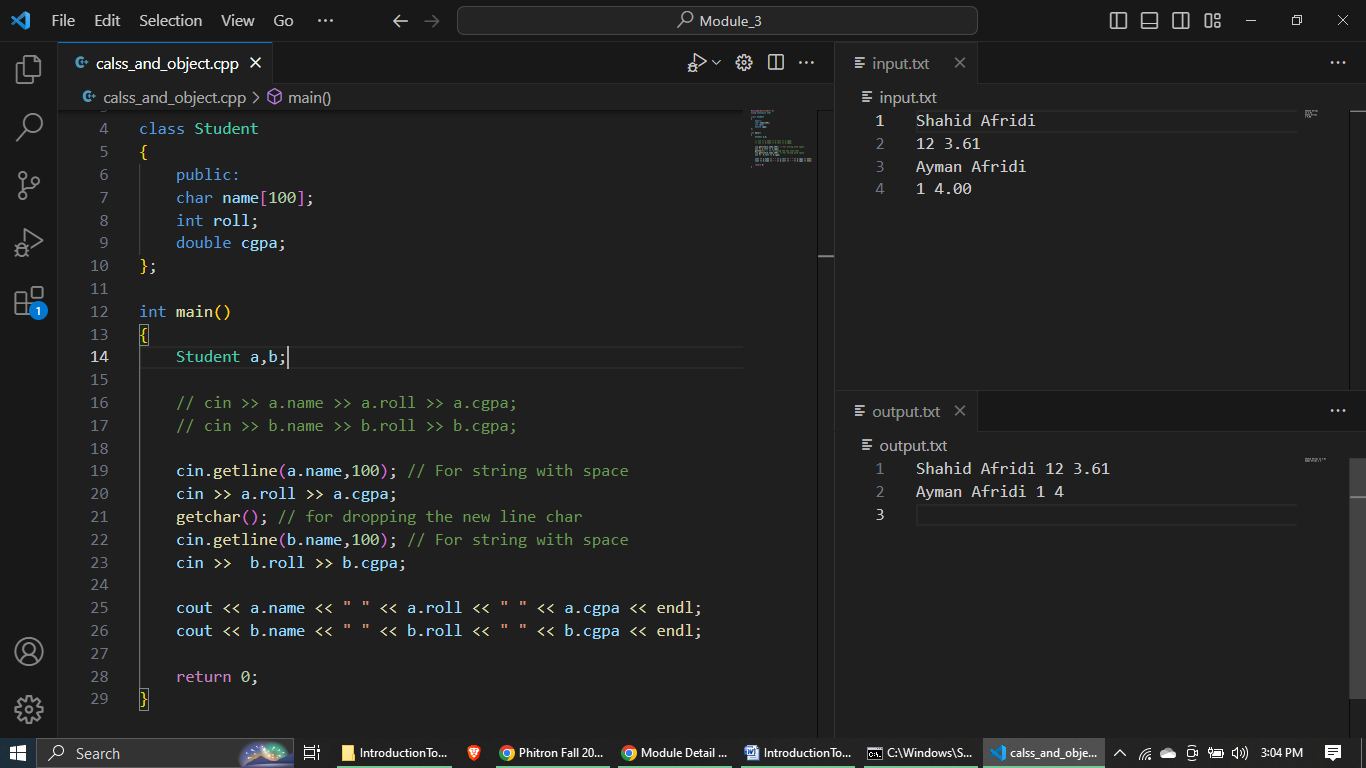
**MODULE - 3 : Class and Object:**

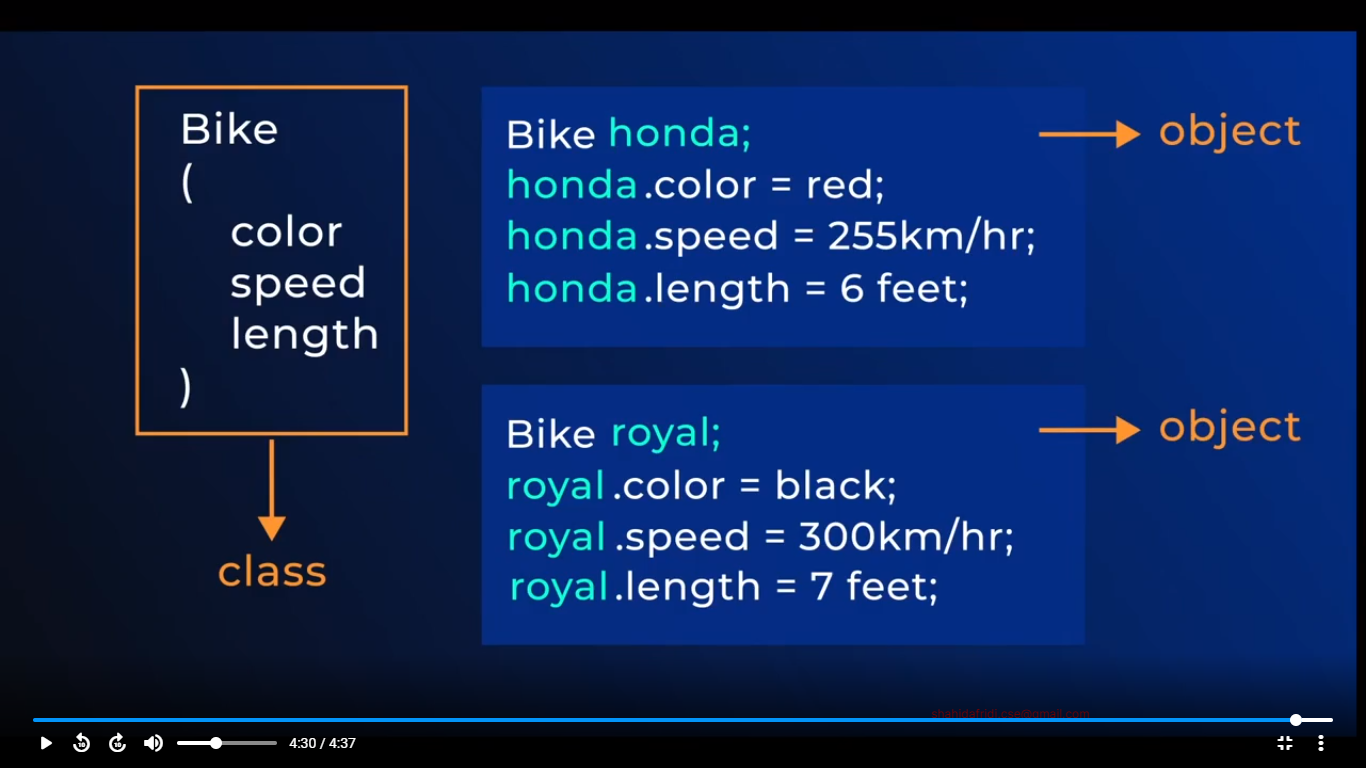
**:**





****

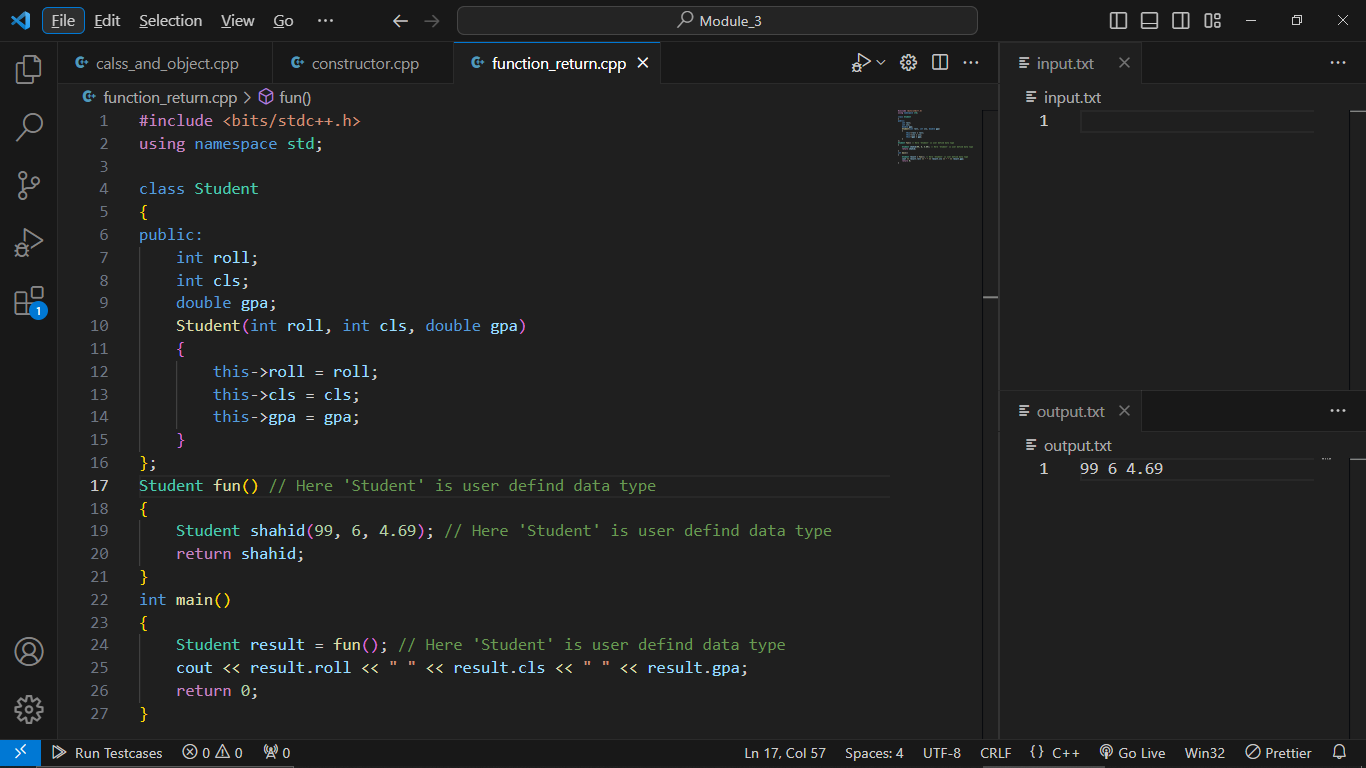
****



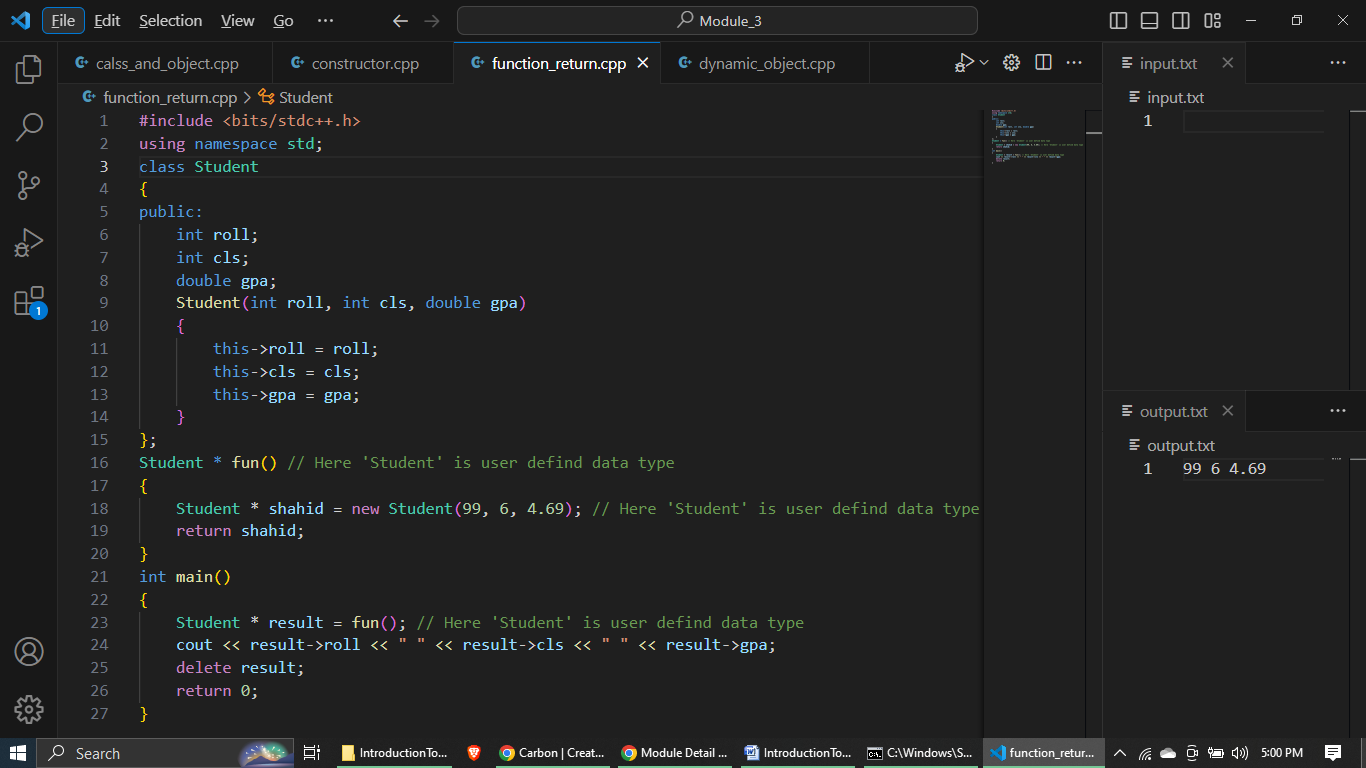
Constructor:

Module - 3-6 Return Object from Function and Why We Need Dynamic Object

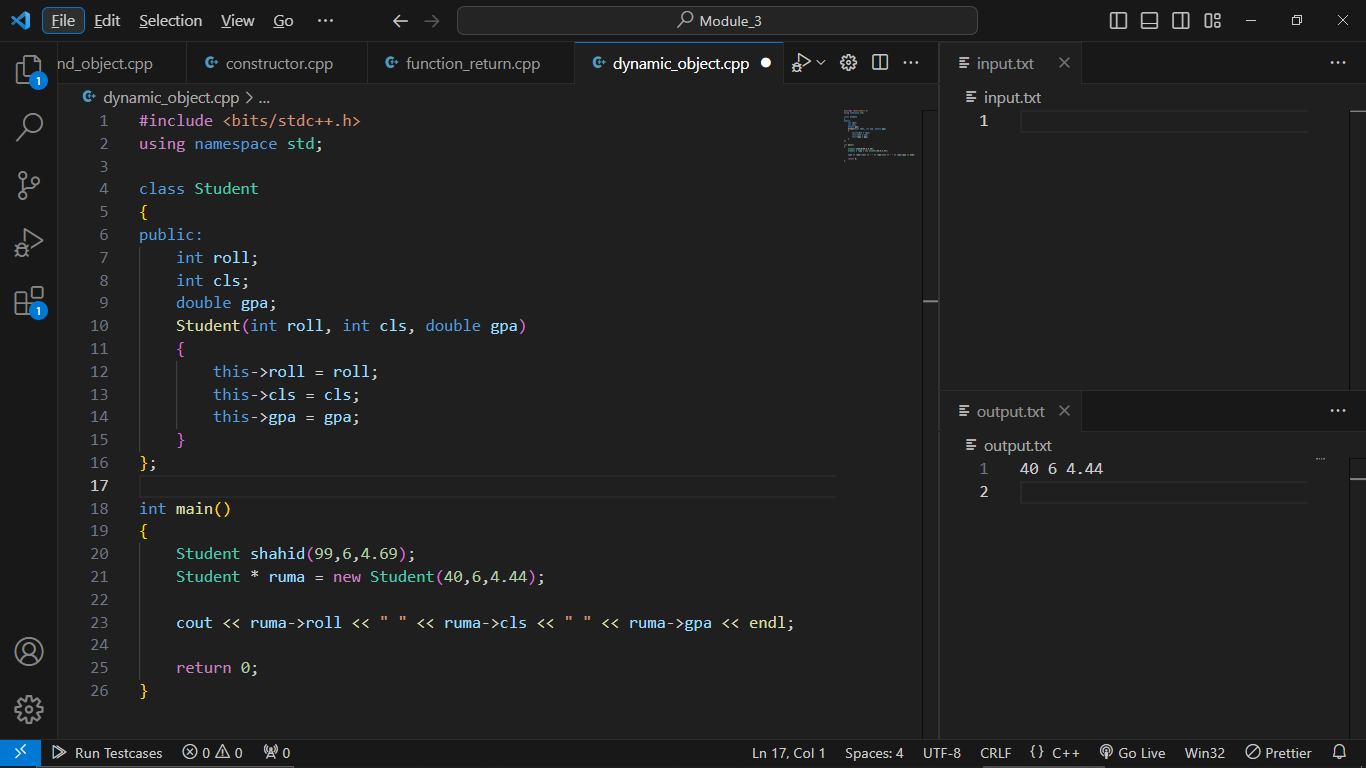
Return Function:



Return Function (Dynamic object pointer):



Dynamic Object:



Sort Array using built-in function:

