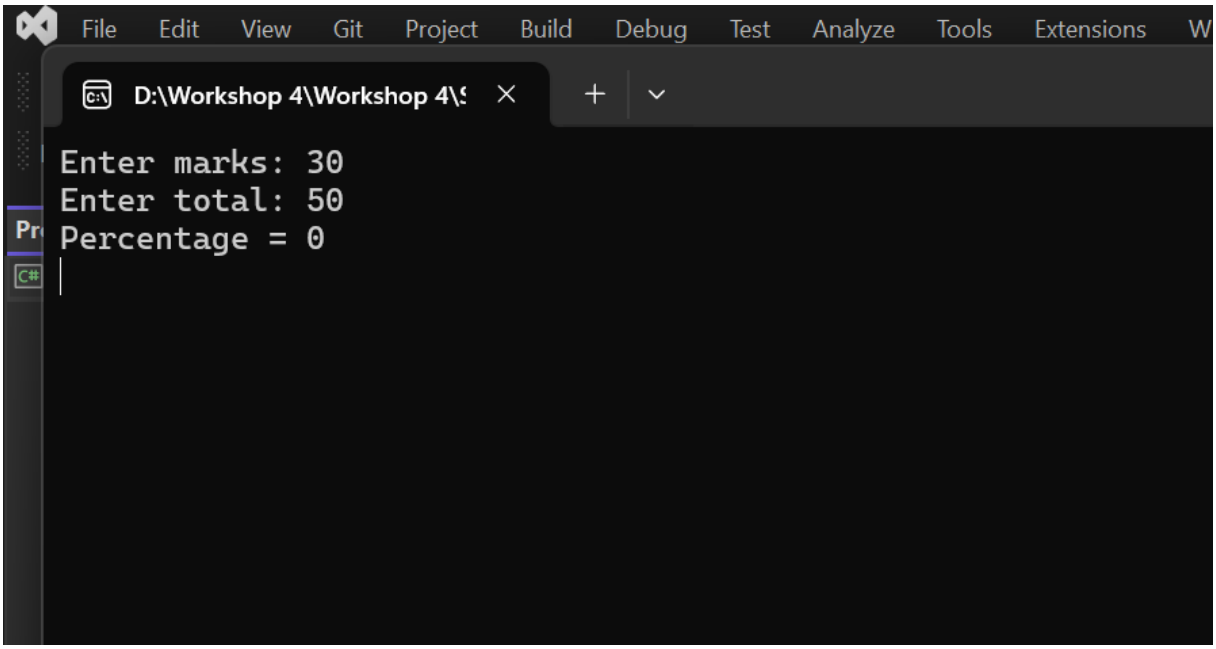
1. Task 6

Here is the program output shown in the screenshots. When I entered the marks and total, the program displayed:

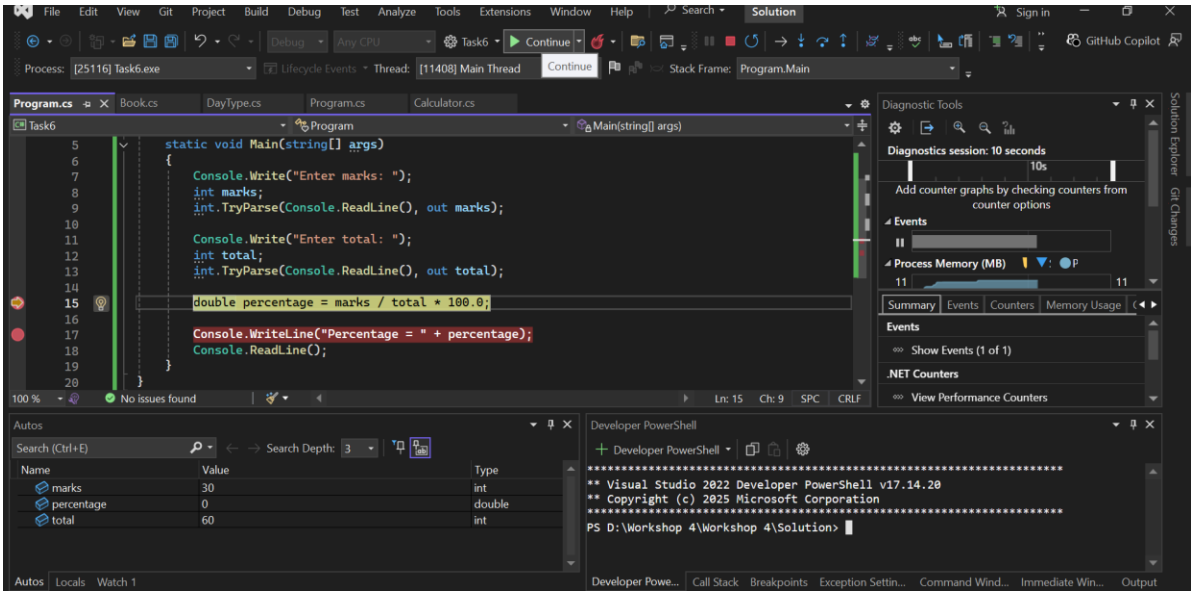


The output became 0, even though the correct percentage should be 60% (because 30 out of 50 is 60%).

Why the output was wrong

The issue happened because the program used integer division.

In the code:



int percentage = marks / total * 100;

Since both marks and total are integers, the division 30 / 50 becomes 0 because integer division removes the decimal value.

So the calculation becomes:

0 * 100 = 0

That's why the percentage printed in the console was 0.

To fix the error, I changed the calculation so that it uses double instead of integer.

I converted the input marks into a double value:

double percentage = (double)marks / total * 100;

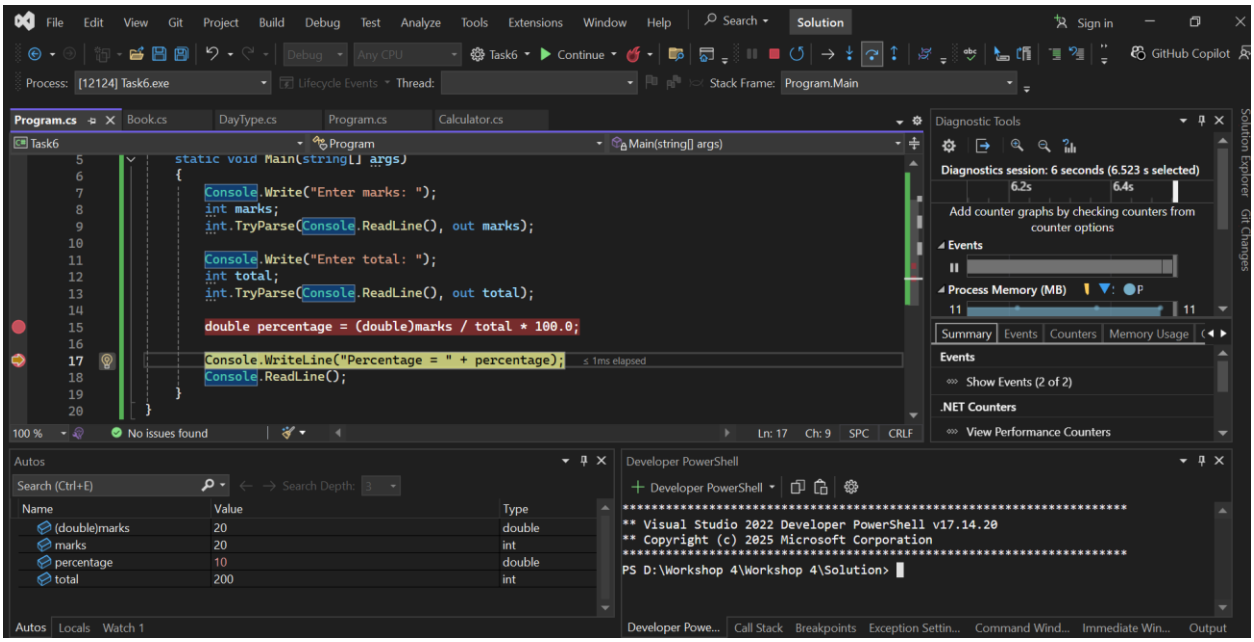
Result after fixing

As shown in the screenshot, the program correctly displayed the percentage value after the type conversion was added.

The debugger also showed that:

- marks was stored as double
- The percentage calculation was correct
- The output printed the right percentage

This solved the problem and the percentage was retrieved successfully.



Task 7

How constructors help in software development

- Constructors are special methods in a class that run automatically when an object is created. Their main job is to make sure the object starts with proper and valid values so the program does not crash or behave incorrectly.

1. Helps During Object Creation

Constructors allow us to set starting values for the object.

This prevents empty, incorrect, or random data inside the object.

Example:

In a banking system, when creating a new Account object, the constructor can set the starting balance to 0, so the account is always valid.

2. Improves Code Reliability

When we use parameterised constructors, we can force the user to give important values.

This avoids runtime errors caused by missing information.

Example:

A Student class may require name and rollNumber in its constructor.

This makes sure a student object is never created without these required values.

3. Makes the Code Easier to Maintain

All the initialization logic stays inside the constructor.

So if we need to change how an object starts, we only update the constructor instead of changing the entire program.

This makes the system easier to update and understand.

Real-Life Use Cases

1. Game Development

Whenever a new Player object is created in a game, the constructor can set default values like:
health = 100
level = 1
starting position on the map
This makes sure every player begins the game properly.

2. E-commerce System
When a user adds a product to the cart, the constructor sets:
product name
price
quantity
This prevents errors such as adding an item without a price.

3. Hospital Management System
When creating a Patient object, the constructor sets:
a unique patient ID
date of admission
an empty list for medical history

This keeps all patient records consistent and complete.