# Part01

* Problem: Write a program to:
  + Accept a string input from the user.
  + Convert it to an integer using both int.Parse and Convert.ToInt32.
  + Handle potential exceptions using a try-catch block.
* Question: What is the difference between int.Parse and Convert.ToInt32 when handling null inputs?

Int.parse: throughs exception.

Convert.ToInt32: returns 0

* Problem: Write a program that:
  + Prompts the user to input a number.
  + Uses int.TryParse to check if the input is a valid integer.
  + valid, print the number; otherwise, print an error message.
* Question: Why is TryParse recommended over Parse in user-facing applications?

To handle a possible exception.

* Problem: Implement a program to:
  + Declare an object variable.
  + Assign it different data types (e.g., int, string, double).
  + Print the GetHashCode() of each assignment.
* Question: Explain the real purpose of the GetHashCode() method.

To identify if tow objects/pointers points to the same address in the memory.

* Problem: Demonstrate how changing one reference affects another when both point to the same object. Use the following steps:

o Create an object and assign it a value.

* + Create a second reference to the same object.
  + Modify the value of the object using one reference and print the value using the other.
* Question: What is the significance of reference equality in .NET?

checks whether two variables refer to the same object

* Problem: Write a program to:
  + Declare a string and modify it by concatenating additional text “Hi Willy”.
  + Print the GetHashCode() before and after modification.
* Question: Why string is immutable in C#?

Because it’s origin is an array of characters with specific length.

* Problem: Create a program to:
  + - Use StringBuilder to append text to a string “Hi Willy”.
    - Print the GetHashCode() of the StringBuilder instance before and after the modification.
* Question: How does StringBuilder address the inefficiencies of string concatenation?

Because it’s origin is a linked list of characters so you can modify it at the same address using append.

* Question: Why is StringBuilder faster for large-scale string modifications?

String: is immutable each modification allocates new object (multiple memory allocation).

StringBuilder: Mutable and allocates only one object.

* Problem: Create a program to:
  + - Accept two integer inputs from the user.
    - Display the sum using all three methods “Sum is input1+input2”:
      * Concatenation (+ operator)
      * Composite formatting (string.Format)
      * String interpolation ($)
* Question: Which string formatting method is most used and why?

String interpolation, because of the clarity and readability.

* Problem: Create a program using StringBuilder to:
  + - Append text.
    - Replace a substring.
    - Insert a string at a specific position.
    - Remove a portion of text.
* Question: Explain how StringBuilder is designed to handle frequent modifications compared to strings.

StringBuilder is optimized for speed and memory efficiency when modifying strings repeatedly, because it modifies a single buffer in-place, unlike immutable string objects

# Part02

1. LinkedIn article about string immutability



1. What’s Enum data type, when is it used? And name three common built\_in enums used frequently?

Enumeration, it’s a data type used to define set of names , makes the code more readable

1. what are scenarios to use string Vs StringBuilder?

String: for small number of operations, simple formatting, constants, code readability.

StringBuilder: Frequent modification, loops, large text.

# Part03 Bonus

1. self study report
2. what meant by user defined constructor and its role in initialization

A user-defined constructor is a special method you create inside a class to initialize objects with specific values at the time of creation.

1. compare between Array and Linked List

Array:

* Index-based, fixed length
* Elements stored in continuous memory

LinkedList:

* Made up of nodes, each with a value and pointer to the next (and optionally previous) node
* Length can be modified.
* Slower to access by position