Part 2

2- What's Enum data type, when is it used? And name three common built_in enums used frequently?

It is something like array but it is constant, you cannot add or modify at runtime and the values inside it are constant and inside it the values stored with value, first is 0, second is 1, ..etc and at compile time the C# converts the numbers to their actual values

It is used to store constants like Days, Status codes and to improve readability

three common built_in enums:

ConsoleColor : for text/background colors in console apps

DayOfWeek: represents days of the week

Environment.SpecialFolder: used to access special system folders

3- what are scenarios to use string Vs StringBuilder?

Use string when the text is fixed and will not change too much to avoid unreachable objects in the heap when modify the object.

Use StringBuilder when the text will modify a lot.

Part 3 (Bonus)

What is the .rdata section ?

It is a ReadOnly section in the memory in the compiled code, it store metadata constants, format strings.

In composite formatting the string stored once in r.data and the other arguments live in the stack

In interpolation, compiler first converts it to string.format and then the string stored in r.data and the same process continues, so interpolation more readable to the developer but the compiler treat as the string.format

How the jump happen in the switch ?

It different in the algorithms based on the type of switch, number of cases

When the values are along each other (dense): case 1, case 2, case 3, ...etc , the cpu does direct jump to the case by using a pointer to the value directly, it will take O(1)

When the values far from each other (Sparse): case 1, case 100, case 100

The cpu does comparisons to reach to the value, it is like if so it will be O(n)

When the values are strings, the compile get the hashcode of strings and make switch on it

The Switch Evolution in the C# versions:

C# 1.0 - 5.0 (Classic switch)

- Works with: int, enum, char, string
- Case labels must be constants
- No ranges, no conditions
- Break required

C# 6.0 (Minor improvements)

- nameof() support

C# 7.0 (Big turning point)

- Switch can now match types
- Introduced pattern variables

C# 7.1 – 7.3

- When conditions supported in the case

C# 8.0

- No break
- Always returns a value
- Cleaner, functional style

C# 9.0

- Relational patterns
- Logical patterns
- not pattern

C# 10.0 (Cleaner syntax)

- Improved readability
- Better lambda + switch integration
- Fewer parentheses

C# 11.0

- Switch can match array and list shapes