1. What is Hashing?

Hashing is the process of converting a value (like a string or object) into a fixed-size number (hash code).

Why it matters:

- Speeds up lookups
- Helps compare objects
- Used in Dictionary, HashSet, etc.

Example:

```
string name = "Ahmed";

Console.WriteLine(name.GetHashCode());
```

---

2. Optimize Memory + Compare

Memory Optimization

C# uses **string interning**: it stores only one copy of identical strings to save memory.

Comparison

- == compares **values**
- ReferenceEquals() compares **memory addresses**

Example:

```
string a = "Hi";

string b = "H" + "i";

Console.WriteLine(ReferenceEquals(a, b)); // True
```

---

3. .rdata Section + switch Evolution

.rdata

In C/C++, .rdata is a section for read-only data (like constant strings).
C# handles constants similarly under the hood for optimization.

switch in C#

- Old style: classic switch-case
- C# 7+: added **pattern matching**
- C# 8+: **switch expressions** — cleaner and shorter

Example:

```
string role = "admin";
string message = role switch {
  "admin" => "Welcome Admin",
  _ => "Access Denied"
};
```

---

4. Why C# Has the Upper Hand

- Modern features: async, LINQ, pattern matching
- Runs on Windows, Linux, macOS (.NET Core & .NET 6+)
- Faster than Python, simpler than C++, cleaner than Java
- Great for: desktop apps, APIs, web, game dev (Unity)