Who developed these languages:

- **C** → Developed by **Dennis Ritchie**
- **C++** → Developed by **Bjarne Stroustrup**
- **Java** → Developed by **James Gosling**
- **Object-Oriented Programming (OOP)** → **Alan Kay**

☞ পরীক্ষার জন্য এক লাইনে মনে রাখার ট্রিক:
C – Dennis, **C++ – Bjarne**, **Java – Gosling**

C language was invented in:
☞ **Bell Laboratories (AT&T Bell Labs)**

Java is developed by
☞ Sun Microsystems of USA

Java compiler translates
☞ source code into Bytecode (Virtual Machine Code)

Java interpreter translates
☞ Bytecode (Virtual Machine Code) into machine code.

নিশ্চয়ই ☺
আমি **OSI Model** টা **A–Z খুব সহজ ভাষায়**, একদম **বাংলায়** বুঝিয়ে দিচ্ছি।

◆ OSI Model কী?

OSI (Open Systems Interconnection) Model হলো কম্পিউটার নেটওয়ার্কে
☞ **ডাটা কীভাবে এক কম্পিউটার থেকে আরেক কম্পিউটারে যায়**
এই পুরো প্রক্রিয়াকে বোঝানোর জন্য বানানো **৭টি ধাপ (Layer)**।

সহজভাবে বললে—
ডাটা পাঠানোর ৭টা ধাপের সিঁড়ি

□ OSI Model এর ৭টি Layer (নিচ থেকে উপরে)

একটা সহজ ট্রিক মনে রাখার জন্যঃ
☞ **"Please Do Not Throw Sausage Pizza Away"**

Layer No	Layer Name
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

□ Layer 1: Physical Layer

☞ কাজ কী?

- **তার, ক্যাবল, সিগন্যাল** নিয়ে কাজ করে
- 0 আর 1 পাঠায়

📌 উদাহরণ:

- LAN cable
- Fiber cable
- Electrical signal

☞ মানুষের ভাষায়:

“ডাটা যাওয়ার রাস্তা বানায়”

□ Layer 2: Data Link Layer

☞ কাজ কী?

- ডাটা **Frame** বানায়
- **MAC Address** ব্যবহার করে
- Error চেক করে

📌 উদাহরণ:

- Switch
- MAC Address

☞ সহজ করে:

“ঠিক বাড়িতে চিঠি পৌঁছাচ্ছে কিনা দেখে”

□ Layer 3: Network Layer

☞ কাজ কী?

- **IP Address** ব্যবহার করে
- কোন রাস্তা দিয়ে ডাটা যাবে সেটা ঠিক করে

📌 উদাহরণ:

- Router
- IP Address

☞ সহজ করে:

“Google Map দেখে সঠিক রাস্তা ঠিক করে”

⚷ Layer 4: Transport Layer

☞ কাজ কী?

- ডাটা ভাগ করে পাঠায় (Segmentation)
- ঠিকঠাক পৌঁছাচ্ছে কিনা দেখে

📌 Protocol:

- TCP (Secure)
- UDP (Fast)

☞ সহজ করে:

“পার্সেল ভেঙে পাঠিয়ে সব ঠিকঠাক পৌঁছাল কিনা দেখে”

🏆 Layer 5: Session Layer

☞ কাজ কী?

- দুই কম্পিউটারের মধ্যে **connection তৈরি, চালু ও বন্ধ** করে

📌 উদাহরণ:

- Video call session
- Login session

☞ সহজ করে:

“কথোপকথন শুরু আর শেষ করে”

🎨 Layer 6: Presentation Layer

☞ কাজ কী?

- ডাটা **Encrypt / Decrypt**
- ডাটা **Compress**
- Format বদলায়

☞ সহজ করে:

“ডাটাকে সাজিয়ে-গুছিয়ে পাঠানোর মতো করে”

Layer 7: Application Layer

☞ কাজ কী?

- ইউজার এখানে কাজ করে
- সফটওয়্যার সরাসরি ব্যবহার করে

📌 উদাহরণ:

- Browser (Chrome)

- Email
- WhatsApp

👉 **সহজ করে:**

“যেখানে আমরা কাজ করি”

একদম ঠিক আছে ☺
আমি ধরেই নিচ্ছি তুমি **একেবারে শুরু থেকে** শিখছো।
চলো **Two-Tier** আর **Three-Tier Database Architecture** একদম **পানি মতো সহজ বাংলায়** বুঝি।

🏗️ Database Architecture মানে কী?

সহজ ভাষায়—

👉 **User (আমরা)**
👉 **Application (Software)**
👉 **Database (Data রাখার জায়গা)**

এই ৩টার মধ্যে **কে কার সাথে কীভাবে কথা বলে**, সেটাই Database Architecture।

🟦 Two-Tier Database Architecture (2 ধাপ)

◆ Two-Tier মানে?

👉 **২টা স্তর (Layer)**

```
User / Application  ↔  Database
```

🧠 এখানে কী হয়?

- User সরাসরি **Application** ব্যবহার করে
- Application সরাসরি **Database** এর সাথে কথা বলে

👉 **কোনো মাঝখানের server নেই**

📌 বাস্তব উদাহরণ:

ধরো—

- তুমি একটা **Desktop Software** ব্যবহার করছো
- সেই Software নিজেই Database-এ গিয়ে data নেয়

☞ যেমন:

- MS Access
- Small Office Software

🖼 কল্পনা করো:

```
[User + Software]  -------->  [Database]
```

✅ সুবিধা:

✓ বানানো সহজ
✓ ছোট কাজের জন্য ভালো
✓ দ্রুত কাজ করে

❌ অসুবিধা:

❌ Security কম
❌ অনেক user হলে slow
❌ বড় system এর জন্য ভালো না

□ Three-Tier Database Architecture (৩ ধাপ)

◆ Three-Tier মানে?

☞ ৩টা স্তর (Layer)

```
User → Application Server → Database
```

□ এখানে কী হয়?

- User সরাসরি Database-এ যেতে পারে না
- মাঝখানে থাকে **Application Server**
- সব rule, security, calculation এখানে হয়

📌 বাস্তব উদাহরণ:

তুমি যখন—

- Facebook ব্যবহার করো
- Banking Software ব্যবহার করো

☞ তুমি Database দেখো না
☞ সবকিছু server এর মাধ্যমে হয়

🖼 কল্পনা করো:

```
[User] → [Application Server] → [Database]
```

✅ সুবিধা:

✓ অনেক বেশি secure
✓ অনেক user handle করতে পারে
✓ বড় company / bank এ ব্যবহার হয়

❌ অসুবিধা:

❌ বানাতে জটিল
❌ খরচ বেশি

□ Two-Tier vs Three-Tier (সহজ টেবিল)

বিষয়	Two-Tier	Three-Tier
Layer সংখ্যা	2	3
Database access	Direct	Server দিয়ে
Security	কম	বেশি
User সংখ্যা	কম	বেশি
ব্যবহার	ছোট software	বড় system

🎯 এক লাইনে মনে রাখার ট্রিক

☞ **Two-Tier:**

User সরাসরি Database-এ যায়

☞ **Three-Tier:**

User → Server → Database

□ একদম বাস্তব উদাহরণ দিয়ে শেষ করি

🏪 দোকানের হিসাব (Two-Tier)

- দোকানদার নিজেই খাতা লেখে
- নিজেই হিসাব দেখে

🏦 ব্যাংক সিস্টেম (Three-Tier)

- Customer → Bank Officer → Vault
- Customer সরাসরি Vault এ যেতে পারে না

একদম ঠিক আছে ☺
আমি **একদম ধাপে ধাপে, খুব সহজ বাংলায়** বুঝিয়ে দিচ্ছি।

◆ দেওয়া আছে:

- **Class:** C
- **Subnet Mask: 255.255.255.224**
  (প্রশ্নে 225 লেখা আছে, কিন্তু **সঠিক Subnet Mask হবে 255**, 225 ভুল)

◆ Step 1: Class C মানে কী?

Class C IP তে:

- মোট **8 bit** থাকে Host এর জন্য

Default Subnet Mask:

```
255.255.255.0
```

◆ Step 2: Subnet Mask কে Binary তে নেই

255.255.255.224 এর শেষ অংশ (224) কে Binary করলে:

```
224 = 11100000
```

☞ এখানে:

- **1 আছে = 3টা**
- **0 আছে = 5টা**

◆ Step 3: Host bit কতটা?

Class C তে মোট host bit = **8 bit**

Subnet Mask এ:

- 3টা bit network নিয়ে নিয়েছে
- বাকি **5টা bit** host এর জন্য

◆ Step 4: Host সংখ্যা বের করি

সূত্র:

```
2^Host bit - 2
```

☞ এখানে Host bit = **5**

```
2^5 - 2
= 32 - 2
= 30
```

□ Final Answer:

✅ 30 users (hosts) possible

□ এক লাইনে মনে রাখার ট্রিক:

224 মানে 11100000 → 5টা zero → $2^5 - 2 = 30$ user

📝 Exam Friendly Answer:

For a Class C IP with subnet mask **255.255.255.224**,
the number of possible users is **30**.

চাও তো আমি
✓ Subnet range
✓ Network & Broadcast address
✓ MCQ practice

সবও খুব সহজ করে বুঝিয়ে দিতে পারি ☺

নিশ্চয়ই ☺
Network Topology খুব সহজ ভাষায়, ধাপে ধাপে বুঝিয়ে দিচ্ছি।

◆ Network Topology কী?

Network Topology মানে হলো—
☞ নেটওয়ার্কে থাকা কম্পিউটার, সার্ভার, সুইচ, কেবল ইত্যাদি **কিভাবে একে–অপরের সাথে যুক্ত আছে তার নকশা বা বিন্যাস**।

সহজ করে বললে ☞ **নেটওয়ার্কের ম্যাপ**।

◆ Network Topology এর ধরন (খুব সহজভাবে)

1 Bus Topology

🚌 বাসের মতো একটাই রাস্তা

কিভাবে কাজ করে:
সব কম্পিউটার **একটি প্রধান কেবল (Backbone)** এর সাথে যুক্ত থাকে।

```
PC ── PC ── PC ── PC
```

✓ সুবিধা:

- কম কেবল লাগে
- খরচ কম

✕ অসুবিধা:

- মূল কেবল নষ্ট হলে পুরো নেটওয়ার্ক বন্ধ
- ট্রাফিক বেশি হলে ধীর হয়

📌 ব্যবহার: ছোট নেটওয়ার্কে (এখন প্রায় ব্যবহার হয় না)

---

2 Star Topology ⋆

⭐ **সবাই এক কেন্দ্রে যুক্ত**

কিভাবে কাজ করে:
সব কম্পিউটার একটি **Hub / Switch** এর সাথে যুক্ত থাকে।

```
      PC
       |
PC — Switch — PC
       |
      PC
```

✅ সুবিধা:

- একটি PC নষ্ট হলেও অন্যগুলো চলবে
- সমস্যা খুঁজে পাওয়া সহজ

❌ অসুবিধা:

- Switch নষ্ট হলে পুরো নেটওয়ার্ক বন্ধ
- কেবল বেশি লাগে

📌 ব্যবহার:
অফিস, স্কুল, বাসা — **সবচেয়ে বেশি ব্যবহৃত**

3️⃣ Ring Topology 🔄

🔄 রিং বা বৃত্তের মতো

কিভাবে কাজ করে:
প্রতিটি কম্পিউটার দুই পাশের কম্পিউটারের সাথে যুক্ত থাকে।

```
PC — PC
|     |
PC — PC
```

ডাটা একদিকে ঘুরে ঘুরে যায়।

✅ সুবিধা:

- ডাটা কনফ্লিকশন কম

❌ অসুবিধা:

- একটি কম্পিউটার নষ্ট হলে পুরো নেটওয়ার্ক বন্ধ
- রক্ষণাবেক্ষণ কঠিন

📌 ব্যবহার: খুব কম (পুরোনো সিস্টেম)

4️⃣ Mesh Topology 🕸️

🕸️ সবাই সবার সাথে যুক্ত

কিভাবে কাজ করে:
প্রতিটি কম্পিউটার অন্য সব কম্পিউটারের সাথে সংযুক্ত।

```
PC ↔ PC
↕  X  ↕
PC ↔ PC
```

✅ সুবিধা:

- খুব নিরাপদ
- একটি পথ নষ্ট হলেও অন্য পথ দিয়ে ডাটা যাবে

❌ অসুবিধা:

- খরচ অনেক বেশি
- সেটআপ জটিল

📌 ব্যবহার:
ব্যাংক, মিলিটারি, গুরুত্বপূর্ণ নেটওয়ার্ক

5️⃣ Tree Topology 🌳

🌳 গাছের মতো কাঠামো

কিভাবে কাজ করে:
একটি মূল নোড থেকে শাখা–প্রশাখার মতো নেটওয়ার্ক ছড়ায়।

```
      Server
        |
    Switch
    /    \
```

```
PC          PC
```

✅ সুবিধা:

- বড় নেটওয়ার্ক সহজে ম্যানেজ করা যায়

❌ অসুবিধা:

- মূল নোড নষ্ট হলে বড় অংশ বন্ধ

📌 ব্যবহার: বড় অফিস, বিশ্ববিদ্যালয়

6️⃣ Hybrid Topology 🔀

🔀 একাধিক topology এর মিশ্রণ

যেমন:
👉 Star + Bus
👉 Star + Ring

✅ সুবিধা:

- ফ্লেক্সিবল
- বড় নেটওয়ার্কে উপযোগী

❌ অসুবিধা:

- ডিজাইন ও রক্ষণাবেক্ষণ জটিল

📌 ব্যবহার: বড় কোম্পানি, ISP

◆ এক নজরে তুলনা (সহজ)

Topology	সহজ?	খরচ	ব্যবহার
Bus	খুব সহজ	কম	ছোট
Star	সহজ	মাঝারি	সবচেয়ে বেশি
Ring	মাঝারি	মাঝারি	কম

Topology	সহজ?	খরচ	ব্যবহার
Mesh	কঠিন	বেশি	গুরুত্বপূর্ণ
Tree	মাঝারি	বেশি	বড় নেটওয়ার্ক
Hybrid	কঠিন	বেশি	বড় প্রতিষ্ঠান

যদি চাও ☺
☞ **চিত্রসহ**,
☞ **MCQ পরীক্ষার জন্য নোট**,
☞ অথবা **real-life example** দিয়ে বুঝিয়ে দিতে পারি।

বললেই হলো 👆

Complexity:

Best, Normal and Worst

Linear → 1 n n

Binary → 1 logn logn

Bubble → n n^2 n^2

Insertion → n n^2 n^2

Selection → n^2 n^2 n^2

Merge → nlogn nlogn nlogn

Heap → nlogn nlogn nlogn

Quick → nlogn nlogn n^2

BFS/DFS → V+E V+E V+E

Dijkstra → E+VlogV E+VlogV E+VlogV

Kruskal → ElogE ElogE ElogE

Prim → ElogV ElogV ElogV

Breadth First Traversal (BFS) এ
☞ আগে যে node ঢোকে, **সেটাই আগে বের হয়**
☞ অর্থাৎ **FIFO (First In First Out)** নিয়মে কাজ করে

আর FIFO নিয়ম মেনে চলে ☞ **Queue**।

□ মনে রাখার ট্রিক:

- **BFS = Queue**
- **DFS = Stack / Recursion**

Items in a priority queue are entered in a _____ order a) random

---

A data structure in which elements can be inserted or deleted at/from both the ends but not in the middle is? a) Dequeue

---

ঠিক আছে 👆
এবার আমি একদম **গোছানো "সূত্র / নিয়ম" আকারে** লিখে দিচ্ছি, যেন খাতায় লিখে মুখস্থ করা যায়।

🔻 Binary, 1's Complement, 2's Complement – গোছানো সূত্র

✅ 1️⃣ Positive সংখ্যা (+)

সূত্র:

Positive সংখ্যার জন্য **কোনো complement লাগে না**
শুধু **direct binary** লিখলেই হবে।

Example:
[

+34 = 00100010
]

◆ 2️⃣ 1’s Complement

সূত্র:

কোনো binary সংখ্যার **সব bit উল্টালে** যা পাওয়া যায়, সেটাই 1’s complement।

Rule:
0 → 1
1 → 0

Example:
[
34 = 00100010
]
[
1’s\ complement = 11011101
]

◆ 3️⃣ 2’s Complement

সূত্র:

2’s complement = 1’s complement + 1

Steps:
1️⃣ আগে positive সংখ্যার binary লেখো
2️⃣ 1’s complement করো
3️⃣ +1 যোগ করো

Example:
[
-34 = 11011110
]

✖ 4 Negative সংখ্যা (–)

সূত্র:

Negative সংখ্যা represent করতে **সবসময় 2's complement ব্যবহার হয়।**

Example:
[
-34 = 11011110
]

□ এক লাইনের সুপার সূত্র

Positive → Direct Binary
Negative → 2's Complement
2's Complement = 1's Complement + 1

📌 Exam-Ready Short Note (এটা হুবহু লিখতে পারো)

In binary number system, positive numbers are represented by direct binary form.
Negative numbers are represented using 2's complement method.
2's complement is obtained by adding 1 to the 1's complement of a binary number.

চাও তো আমি এটাকে **আরও ছোট "cheat-sheet"**, বা **১ পেজ নোট PDF স্টাইলেও** সাজিয়ে দিতে পারি ☺