Assignment1: Research and present a comparison of different garbage collection algorithms (Serial, Parallel, CMS, G1, ZGC) in Java.

Answer:

1. Serial Garbage Collector (Serial GC)

- **Type:** Stop-the-world (STW), single-threaded.
- **Usage:** Best suited for small applications or environments with limited resources, such as client-side applications or small-scale server applications.
- **Performance:**
  - **Heap Size:** Small.
  - **Pause Time:** Longer pauses, as it stops all application threads during garbage collection.
  - **Throughput:** Lower throughput due to single-threaded nature.
- **Pros:** Simple, easy to implement, low overhead.
- **Cons:** Not suitable for applications with large heaps or those requiring low-latency.

2. Parallel Garbage Collector (Parallel GC)

- **Type:** Stop-the-world (STW), multi-threaded.
- **Usage:** Suitable for mid-sized to large-scale applications running on multi-core machines.
- **Performance:**
  - **Heap Size:** Medium to large.
  - **Pause Time:** Moderate pauses, improved over Serial GC due to multi-threading.
  - **Throughput:** High throughput, as it utilizes multiple threads.
- **Pros:** Good for applications prioritizing throughput.
- **Cons:** Pause times can still be significant, not ideal for low-latency applications.

3. Concurrent Mark-Sweep Garbage Collector (CMS GC)

- **Type:** Concurrent, low-latency.
- **Usage:** Applications where minimizing pause times is crucial, such as interactive or real-time applications.
- **Performance:**
  - **Heap Size:** Medium to large.
  - **Pause Time:** Shorter pauses due to concurrent phases.
  - **Throughput:** Moderate to high.
- **Pros:** Low pause times, concurrent marking and sweeping phases.

- **Cons:** Higher CPU usage, fragmentation issues, potential for longer-term performance degradation due to fragmentation.

4. Garbage-First Garbage Collector (G1 GC)

- **Type:** Region-based, adaptive, low-latency.
- **Usage:** Designed for large heaps (multi-gigabyte) and modern server applications.
- **Performance:**
  - **Heap Size:** Large.
  - **Pause Time:** Low pause times, with goal-oriented pause time targeting.
  - **Throughput:** High throughput, balances pause times and application performance.
- **Pros:** Adaptive, good for large heaps, predictable pause times.
- **Cons:** More complex than CMS, tuning may be required.

5. Z Garbage Collector (ZGC)

- **Type:** Concurrent, low-latency, scalable.
- **Usage:** Large-scale applications requiring very low pause times, especially for applications with heaps in the multi-terabyte range.
- **Performance:**
  - **Heap Size:** Very large.
  - **Pause Time:** Extremely low (sub-millisecond).
  - **Throughput:** High, designed to minimize interference with application threads.
- **Pros:** Very low pause times, scalable, suitable for large heaps.
- **Cons:** Requires more recent JVM versions, more complex implementation.