SESSION-05 TASKS

Task1:

Question: mention examples for programming languages that have garbage collection and mention examples for those that doesn't have it?

Answer:

Garbage collection (GC) is a memory recovery feature built into programming languages . A GC-enabled programming language includes one or more garbage collectors (GC engines) that automatically free up memory space that has been allocated to objects no longer needed by the program.

Unfortunately, garbage collection is both slow and unpredictable.

Slow is always bad if you're building something that you want to run fast - but unpredictable is a more subtle issue.

Garbage collection is a feature in many programming languages that automatically manages memory allocation and deallocation for objects and data structures, which helps to prevent memory leaks and other memory-related errors. Some programming languages that have garbage collection include:

Java

Python

Ruby

C#

JavaScript

Go

Kotlin

PHP

Swift doesn't have a garbage collector. Instead, it uses a much faster technique called *Automatic Reference Counting (ARC)*.In most cases, this means you don't have to worry about memory handling in Swift. It simply works, as if it were a garbage collector. In other words, ARC will automatically deallocate the piece of memory used by a class when it is not going to be used anymore.

Primitive programming languages like C and C++ do not have their garbage collection instead expect the developer to not only allocate the object but also deallocate it explicitly. Hence we see the functions like "malloc" and "free".

On the other hand, there are several programming languages that do not have built-in garbage collection, such as:

C

C++

Assembly

Rust (although it has some support for optional garbage collection)

In these languages, the programmer is responsible for manually managing memory allocation and deallocation, which can be more error-prone and challenging but can also provide more control over the use of system resources.

Task 2:

Question: how to make operating system control garbage collection?

Answer:

The operating system itself generally does not control garbage collection directly, as it is the responsibility of the programming language runtime or virtual machine to implement garbage collection. However, the operating system can have an impact on the performance of garbage collection by affecting the availability of system resources such as memory and CPU.

To optimize garbage collection performance, the operating system can provide features such as:

Memory management: The operating system can provide virtual memory management features to allocate and deallocate memory to the process, which can help to reduce memory fragmentation and improve the efficiency of garbage collection.

Processor scheduling: The operating system can manage the scheduling of processes and threads to ensure that garbage collection runs efficiently without being interrupted by other processes or threads.

Resource allocation: The operating system can allocate system resources such as CPU time, disk I/O, and network bandwidth to the garbage collection process, which can help to reduce the time required for garbage collection.

In addition, the operating system can provide tools and APIs for monitoring and analyzing the performance of garbage collection, which can help programmers to optimize garbage collection in their applications.