Part 2 - Benchmarking Lua's GC

1. **Use the callgrind outputs or call graphs to find the percentage of instructions consumed by GC for the different GC configurations. Report the results and explain the variation in the results.**

a) Full GC :

Percentage of instructions consumed by full GC:

$$\text{Percentage} = \frac{GC\ Instruction\ Count}{Total\ Instruction\ Count} X\ 100$$

$$= \frac{220361720}{1164646834} X\ 100$$

$$= 18.92\ \%$$

b) Incremental GC :

Percentage of instructions consumed by incremental GC:

$$\text{Percentage} = \frac{GC\ Instruction\ Count}{Total\ Instruction\ Count} X\ 100$$

$$= \frac{220362720}{1164647854} X\ 100$$

$$= 18.92\ \%$$

c) Generational GC :

Percentage of instructions consumed by generational GC:

$$\text{Percentage} = \frac{GC\ Instruction\ Count}{Total\ Instruction\ Count} X\ 100$$

$$= \frac{217999852}{1162230801} X\ 100$$

$$= 18.75\ \%$$

2. Testbench on Different Parameter :

Full GC results on different set of parameters:

Parameter Set	GC Instruction Count	Total Instruction Count	(%) of Instruction Count
m=100, n=100	13,341,287	106495270	12.54%
m=500, n=100	93,269,202	592416818	15.75%
m=5000, n=100	1,044,960,871	5921921540	17.64%

Incremental GC results on different set of parameters:

Parameter Set	GC Instruction Count	Total Instruction Count	(%) of Instruction Count
m=100, n=100	13,342,249	106497665	12.53%
m=500, n=100	93,270,100	592416724	15.75%
m=5000, n=100	1,044,961,853	5921923509	17.65%

Generational GC results on different set of parameters:

Parameter Set	GC Instruction Count	Total Instruction Count	(%) of Instruction Count
m=100, n=100	13122971	106279195	12.34%
m=500, n=100	92,244,715	591393285	15.59%
m=5000, n=100	1033376719	5910339751	17.48%

Result Analysis :

The GC overhead is relatively low because fewer objects are managed, leading to fewer GC instructions. As the number of objects increases, the GC workload grows significantly, consuming more instructions. This is due to increased allocation and deallocation of memory. As m increases, the percentage of instructions consumed by GC increases because the garbage collector has more objects to process, resulting in a higher computational cost.

3. Perf Analysis :

Metrics	Full GC	Incremental GC	Generational GC
Branch Misses	575912	619433	720825
Page Faults	15250	11220	10279
Cache Misses	214579	2899110	3367458
Instructions Per Cycle	2.30	1.77	1.58

Branch Miss: It indicates how efficiently the CPU can predict branches during GC operations.

Page Faults: It reflects the memory management overhead introduced by GC.

Cache Misse: It shows the impact of GC on cache utilization and memory access patterns.

Instructions Per Cycle: It measures the efficiency of instruction execution with and without GC.