**LoadRunner Questions and Answers Summary**

### 1. **How to Integrate LoadRunner Enterprise (LRE) with New Relic?**

* **Approach:**
  + Inject custom headers in LoadRunner scripts.
  + Ensure applications monitored by New Relic capture these headers.
  + Use New Relic APM’s transaction traces and logs to correlate user traffic.
  + Some custom integrations require tagging transactions using headers like X-NewRelic-ID or X-Request-ID.

### 2. **Little’s Law**

* **Formula:** L = λ × W
  + **L** = average number of users in the system (concurrent users)
  + **λ** = arrival rate (requests/sec)
  + **W** = response time (seconds)
* **Usage in Performance Testing:**
  + To validate expected user load
  + Identify resource bottlenecks

### 3. **LoadRunner Algorithms**

* **Load Distribution Algorithms:**
  + **By Group** – execute scripts grouped together
  + **By Scenario** – all scripts start simultaneously
  + **Random** – VUsers are picked randomly
* **Pacing Algorithms:**
  + Fixed pacing (e.g., every 10 seconds)
  + Random pacing within range
* **Think Time:**
  + Replay as recorded
  + Multiply/Ignore/Random

### 4. **Regular Expressions (RegEx)**

* Used for **pattern matching** in dynamic values
* Example in LR:

web\_reg\_save\_param\_regexp(  
 "ParamName=token",  
 "RegExp=\"auth\_token\":\"(.\*?)\"",  
 LAST);

* Useful for correlation in JSON/XML responses

### 5. **Atoi and Itoa Functions in LoadRunner**

* atoi(char \*str) – Converts string to integer
* itoa(int value, char \*str, int base) – Converts int to string

int x = atoi("100");  
char str[10];  
itoa(x, str, 10);

### 6. **Web Custom Request Parameters**

Used for highly customizable HTTP requests.

web\_custom\_request("LoginAPI",  
 "URL=https://api.site.com/login",  
 "Method=POST",  
 "Body={\"username\":\"{user}\",\"pass\":\"{pass}\"}",  
 LAST);

### 7. **JVM Architecture**

* **Method Area** – class structure
* **Heap** – runtime data, objects
* **Stack** – per-thread execution
* **PC Register** – current instruction pointer
* **Native Method Stack** – JNI methods
* Managed by **Garbage Collector**

### 8. **How to find RCA in APM (Root Cause Analysis)**

* Use tools like **Dynatrace**, **New Relic**, **AppDynamics**
* Check for:
  + Slow transactions
  + Error rates
  + Thread dumps
  + CPU/memory usage
  + Database calls and external services

### 9. **Header to Send from LR to Dynatrace**

Add web\_add\_header():

web\_add\_header("x-dynatrace-test", "LoadTest=true;VU={vuser\_id}");

Used to tag traffic for correlation in Dynatrace.

### 10. **Garbage Collection (GC) and Algorithms**

* **Serial GC** – Single-threaded, stop-the-world
* **Parallel GC** – Multithreaded, high throughput
* **CMS** – Concurrent Mark-Sweep (deprecated)
* **G1 GC** – Region-based, balances pause and throughput
* **ZGC / Shenandoah** – Ultra-low pause, concurrent

### 11. **Different Functions in LoadRunner**

#### a. Parameter Handling

* lr\_eval\_string(), lr\_save\_string(), lr\_paramarr\_len()

#### b. Transaction Handling

* lr\_start\_transaction(), lr\_end\_transaction()

#### c. Logging

* lr\_output\_message(), lr\_error\_message()

#### d. Think Time

* lr\_think\_time()

#### e. Correlation

* web\_reg\_save\_param(), web\_reg\_save\_param\_json()

#### f. Web Functions

* web\_url(), web\_submit\_data(), web\_custom\_request()

#### g. Cookies

* web\_add\_cookie(), web\_cleanup\_cookies()

#### h. C Conversions

* atoi(), itoa(), sprintf()

#### i. Rendezvous

* lr\_rendezvous()

#### j. Timers

* lr\_start\_timer(), lr\_end\_timer()

*End of Document*