Write a Bash script that monitors a single process for memory leaks and collects diagnostic evidence when memory usage exceeds a specified threshold.
The script must:
* Accept a process name or PID
* Monitor memory usage at regular intervals, accept interval from user
* Detect when memory exceeds a threshold, accept threshold from user
* Collect diagnostic artifacts, accept directory path from user for storing the artifact
* Optionally restart the process, if user wants
* Log all activities with timestamps
* Exit with codes suitable for monitoring/alert systems

1.) Accepting inputs

flags
- pid
- t - threshold
- i - interval
- d - directory
- r - restart

case

getopts

2) Validate inputs - [[-z]]

$#

3) Resolve PID - pgrep

ps -p / ps -ef / /proc/$PID

4) Monitor loop — while
sleep

5) Threshold detection — if [[ ]]

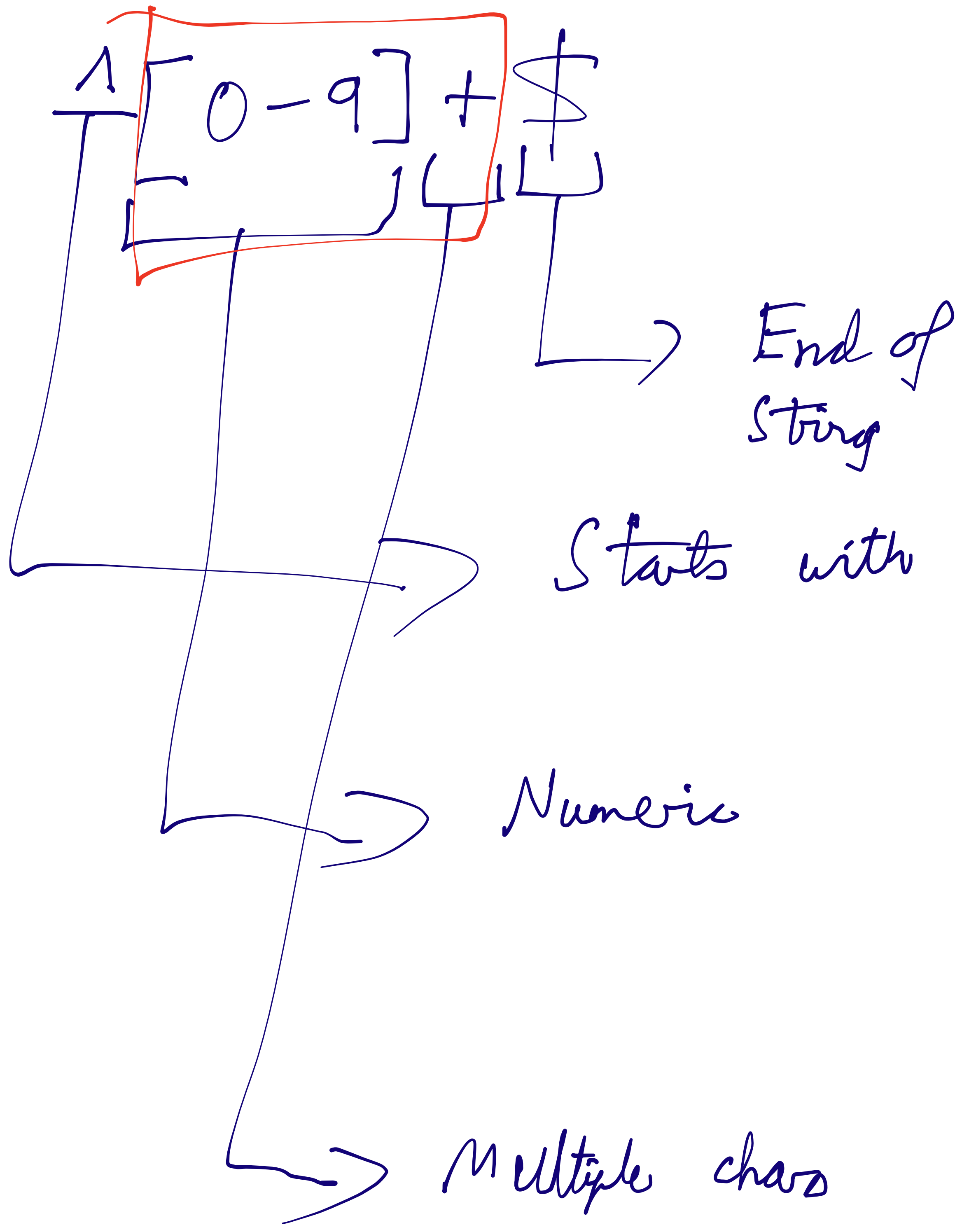
6) Evidence Collection — cp /proc/$PID
$DIR

7) Restart — kill -9

8)
Invalid input — exit code 1
PID inactive — exit code 2

SKELETON

```
# !/bin/bash
set -euo pipefail
trap . . . . .

# ARG INPUT
# ARG VALIDATION

# RESOLVE PID & PID VALIDATION

# while process exists;
    → measure memory usage
    → if threshold exceeds, then get
      evidence

# EVIDENCE COLLECTION
    → create folder
    → copy files from /proc/$PID

    → Restart if needed
```


^[0-9]+$
End of String
Starts with
Numeric
Multiple chars

```
root@ip-172-31-27-45:~# cat diagnotics.sh
#!/bin/bash
set -euo pipefail

THRESHOLD=""
INTERVAL=""
EVIDENCE_DIR=""
RESTART=false
PROCESS=""
PID=""


usage(){
  echo "Usage $0 -p <pid> -t <threshold %> -i <interval_seconds> -d <evidence directory> [-r
optional restart] "
  exit 1
}

while getopts "p:t:i:d:r" opt; do
        case $opt in
                p) PROCESS="$OPTARG" ;;
                t) THRESHOLD="$OPTARG" ;;
                i) INTERVAL="$OPTARG"  ;;
                d) EVIDENCE_DIR="$OPTARG"  ;;
                r) RESTART=true ;;
                *) usage ;;
        esac
done

[[ -z "$PROCESS" || -z "$THRESHOLD" || -z "$INTERVAL" || -z "EVIDENCE_DIR" ]] && usage

[[ ! "$THRESHOLD" =~ ^[0-9]+$ ]] && {echo "Threshold should be numeric "; exit 1}

[[ ! "$INTERVAL" =~ ^[0-9]+$ ]] && {echo "Internval should be numeric "; exit 1}

mkdir -p "$EVIDENCE_DIR" || {echo "cannot write to given directory "; exit 1;}

if [[ "$PROCESS" =~ ^[0-9]+$ ]]; then
        PID="$PROCESS"
else
        PID=$(pgrep -x "$PROCESS")

        #QUESTION REMIDNER


[[ ! -d "/proc/$PID" ]] && {echo "PID $PID not active"; exit 2;}
```

```
while true; do
  [[! -d "/proc/$PID" ]] && {echo "Process died"; exit 2; }

  rss_kb=$(grep "VmRSS:" "/proc/$PID/status" | awk '{print $2}')
  total_kb=$(grep "MemTotal:" /proc/meminfo | awk '{print $2}')
  percent=$(( rss_kb*100/total_kb))
  echo "Current Memory usage of process is $percent%"

  if [[ "$percent" -ge "$THRESHOLD" ]]; then
        echo "Threshold exceeded"
        edir="$EVIDENCE_DIR/$PID-$(date '+%Y%m%d-%H%M%S')"
        mkdir -p "$edir"

        cp "/proc/$PID/maps" "$edir/maps.txt" 2>/dev/null || true
        cp "/proc/$PID/status" "$edir/status.txt" 2>/dev/null || true
        ls -l "/proc/$PID/fd" > "$edir/fd.txt" 2>/dev/null || true

        if $RESTART; then
              echo "Attempting graceful restart"
              kill -15 "$PID" 2> /dev/null || true
              sleep 10

              [[ -d "/proc/$PID" ]] && kill -9 "$PID" 2>/dev/null || true
              echo "restart completed"
        fi
fi
sleep "$INTERVAL"
done
```