Q. 3) Know your system: Find a multicore environment where you can use at least two cores. Investigate the environment and answer the following:

1. What is the model, make and core count of your CPU?
    a. Model – Intel Core i5, 2.9 GHz
    b. Make – i5-5287U
    c. Core count – 2
    d. Thread count – 4

2. What are the L1, L2, L3, (L4?) cache, and memory sizes?
    a. L1, L2 and L3 caches are different memory pools similar to the RAM in a computer. They were built in to decrease the latency to access data by the processor. The architecture they are built with also differs considerably. For e.g. the L1 cache is built using larger transistors and wider metal tracks, trading off space and power for speed. The higher-level caches are more tightly packed and use smaller transistors.
    b. L1 cache – 64 KB (code), 64 KB (data)
    c. L2 cache – 256 KB (per core)
    d. L3 cache – 3 MB

3. Clock speed, the MIPS and the MFLOPS rating for your CPU?
    a. Clock Speed – 2.9 GHz
    b. MIPS and MFLOPS are outdated ways of measuring the speed of a processor, because a processor might be optimized to, let's say ADD and could finish 100 instructions of it really fast but take longer for other commands.

4. Disk seek, latency, and transfer times (your local hard disk)?
    There doesn't exist a hard drive in my computer rather SSD, it doesn't have any disk seek time. However, there is an access time associated with how fast a page of data can be accessed
    a. Read Speed – 1350.7 MB/s
    b. Write Speed – 1411 MB/s

5. How to know if your processor is 64b or 32b? How do you know if the processor supports "hyperthreading"?
    a. We can find out by typing "uname –p" in the terminal, if the output is i386 that means the processor is 32 bits.
    b. Hyperthreading enables a logical view of the processor and presents the physical cores as more to the OS. We can use "sysctl –n hw.ncpu", if there is a difeerence in the number of cores shown, hyperthreading is enabled.

6. OS version? Is 32 or 64 bit?
    a. OS version – macOS Sierra, v10.12.2
    b. It is 64bits

7. Available VM size for user processes. Can it be changed?
    Virtual memory, also known as the swap file, uses part of your hard drive to effectively expand your RAM, allowing you to run more programs than it could otherwise handle.
    a. Its unlimited.
    b. Yes, it can be changed. Only the total memory can be changed, not the memory allotted to each process.

8. Limits on the stack space, heap space and static area. Can they be changed?
    a. Stack Space – 8192 KB
    b. Heap Space – There are limits, exact size couldn.t be found
    c. Static area – Depends on the Bus size and the virtual memory size
    Yes, all of them can be changed using "ulimit –s STACK_SIZE"

9. What is a memory leak and how can you detect one?
    A memory leak is a type of resource leak that occurs when a computer program incorrectly manages memory allocations in such a way that memory which is no longer needed is not released.
    1. Find the memory leak – Detect the presence of a memory leak in the system, given a particular reproducible sequence. You should be able to identify a specific process, but demonstrating an overall increase in committed system memory can qualify a memory leak as well.
    2. Isolate the memory leak – Determine the exact location in the source code where the un–freed allocation occurs. This can be a lengthy and tedious process, requiring specific tools, trial–and–error, and teamwork with the original author of the code.
    3. Fix the memory leak – After the first two steps are completed, this is easy. Fixing the memory leak usually involves adding some code to free the memory in the questionable code path.

10. How can one cause a signal on stack overflow?

```
void sighandler(int signum)
{
   printf("Process %d got signal %d\n", getpid(), signum);
   signal(signum, SIG_DFL);
   kill(getpid(), signum);
}

int main()
{
   signal(SIGSEGV, sighandler);
   printf("Process %d waits for someone to send it SIGSEGV\n",getpid());
   sleep(1000);
   return 0;
}
```

11. How can one tell how many page faults a process had?
    ps –o minflt,majflt pID

12. How can one tell how much user, system and elapsed time a process used?

```
perl -MPOSIX -l -0777 -ne '@f = /\(.*\)|\S+/gs;
   printf "utime: %.2f\nstime: %.2f\ncutime: %.2f\ncstime: %.2f\n",
    map {$_/POSIX::sysconf( &POSIX::_SC_CLK_TCK )}@f[13..16]' "/proc/$pid/stat"
```

13. How can one time a procedure in a program? How accurate is this?
    We can use the "time" utility provided in *nix systems. For e.g. "time -v sleep 1".
    It returns the amount of time sleep worked in the PID 1