Problems for January 25th - Python Math, CPUs, RAM, Storage

Problem A: Storage Budget

In this problem, the goal is to write a program that accepts several inputs and determine how much RAM vs hard-drive storage a person can afford to purchase. The two inputs are

- The monetary budget (in dollars)
- Ther percentage of the budget that is for non-volatile storage (HDD/SSD)

The program should calculate how many gigabytes of non-volatile storage (HDD/SSD) and volatile storage (RAM) the user can afford, based on these to inputs and the fixed values of:

- Non-Volatile storage costs $0.50 per gig
- Volatile storage costs $10.00 per gig

Some examples:

```
Computer Storage budget: 200
% of non-volatile (long-term) storage: 30
you can afford 120.0 gigs of hard-drive storage
you can afford 14.0 gigs of RAM
```

```
Computer Storage budget: 500
% of non-volatile (long-term) storage: 20
you can afford 200.0 gigs of hard-drive storage
you can afford 40.0 gigs of RAM
```

Problem B: CPU Ghz

In this problem, the goal is to write a program that accepts several inputs and determines the overall Ghz performance of a computer (or multi-computer) setup. The three inputs:

- The number of CPUs (an integer number)
- The CPU Ghz (a float)
- The number of cores per CPU (an integer number)

The program should calculate the total number of cycles (operations) that the computer setup should be able to calculate given these three numbers.

Some examples:

```
Enter CPU count: 1
Enter CPU ghz: 3.1
Enter CPU core count: 4
This config has 12.4 total ghz of computational power


Enter CPU count: 3
Enter CPU ghz: 2.5
Enter CPU core count: 8
This config has 60.0 total ghz of computational power
```

Problem C: CPU Ghz (pt 2)

In this problem, the goal is to write a program that accepts the same three inputs as from problem B, but for TWO separate computer setups. Thus, it should take six total input, one for setup A, and another for setup B.

The program should calculate the total number of cycles (operations) for both computer setups, and then compare the two.

- If setup A is better than B the program should print:
  `Configuration A is better than B`
- If setup B is better than A the program should print:
  `Configuration B is better than A`
- If status are equivalently good, then print:
  `Configurations A and B equally as good`

Note that this required using an if-statement, which some students might not have watched the video for yet (video 10, and more will be added for next week). Some examples:

```
A: Enter CPU count: 1
A: Enter CPU ghz: 2.0
A: Enter CPU core count: 2
B: Enter CPU count: 2
B: Enter CPU ghz: 1.0
B: Enter CPU core count: 2
Configurations A and B equally as good
```

```
A: Enter CPU count: 5
A: Enter CPU ghz: 2.1
A: Enter CPU core count: 4
B: Enter CPU count: 1
B: Enter CPU ghz: 3.5
B: Enter CPU core count: 1
Configuration A is better than B
```