**Fall 2020 Computer Science 220**
**Program assignment 2**

**Learning objectives:**

1. Develop a simple Python program that does arithmetic
2. Develop a Python program with a loop, practice accumulating result
3. Develop a Python program with a nested loop

**Assignment:**

**Part I**

A company needs calculating the total working time of employees in a team. You are going to help to write a program that totals the working time in hours and minutes. You should name your program as **`working_time.py`**.

Your program should first ask the user how many employees there are on the team. You may assume the value entered will not be negative, but it could be zero. Your program will now be controlled by a loop that asks for, and adds, the hours and minutes for each employee. Of course, the minutes will frequently add up to more than an hour, but you should not report a total time such as 24 hours 77 minutes for example (or worse, 18 hours 377 minutes). Instead you should report the total time as 25 hours 17 minutes. On the other hand, if the total number of hours is 24 or greater you should **not** report the time in days, hours and minutes. Hours and minutes are all that are to be reported.

While you may assume that all of the hours and minutes entered by the user will be non-negative, a time may be entered with minutes that exceed 60. Your program should be able to handle this and still report the total correctly. For example:

```
employee1: 5  hours 65 minutes
employee2: 3  hours 88 minutes
employee3: 18 hours 45 minutes
---------------------------------------

Total time: 29 hours 18 minutes
```

Note:

1. If you already know about conditionals, you **may not** use them in this program.

To

**Part II**

Do not try to do Parts I and II all together. Get Part I working completely correctly, and then go on to Part II.

In part II, the company needs to calculate the total working time for employees from all teams in days, hours, and minutes. Users of the program report that they do not like having to start the program over and over to process multiple teams, so we are going to fix that complaint.

Add code to first ask how many teams are to be processed (you may assume the value entered is non-negative). Use this number to determine how many times the program will go through one loop (known as an *outer* loop) that contains within it the code from part I. Since the loop from the code of part I will now lie within the loop you are developing for this part of the assignment, it is known as the *inner* loop. At the end of each pass of the

outer loop your program should identify the team being processed as Team1, Team2, etc. and then report the total time of that Team. Thus, we may see output such as:

```
Team1 Total time: 29 hours 18 minutes
Team2 Total time: 27 hours 49 minutes
Team3 Total time: 15 hours 1 minutes
```

(It's OK to use the plural form "minutes" in this case.)

Note that the first team is identified as Team1, not Team0.

Finally, add code to output the total time of all of the teams entered. Thus, the last line of output may be:

```
Total time of all teams: 3 days 0 hours 8 minutes
```

Note that the total time for all teams **should** display the number of days, hours, and minutes.

**Documentation and Style:** Follow the instructions posted in homework policy.

.

**Submission:** Submit `working_time.py` to your class OAKS