Workshop 1 - Environment and Language Familiarisation.

We will be using a Bash shell on a Linux server and Python to develop our apps. This module is a course on neither. Many of you will already by familiar with Bash and/or Python, if not, this week is your opportunity to catch up.

If you are not familiar with Bash, this is a good tutorial: *http://mywiki.wooledge.org/BashGuide*

If you are not familiar with Python, this is a good tutorial: *http://pythonprogramminglanguage.com/*

Spend this workshop becoming familiar with these two languages using the above tutorials. The following tasks can both be completed in Bash and Python. Pick the language you are least familiar with and develop solutions for both using the above tutorials (and of course, skilful googling) to teach yourself the language.

Task 1: Factorials

The factorial of a number is a multiple of all integers between 1 and that number inclusive. For example, the factorial of 5 (expressed as 5!) would be 5x4x3x2x1 = 120.

Write a program that calculates the factorial of each input value. Input data

The input will consist of a series of positive integers between 1 and 15 inclusive, each on a line by itself. The end of the input data file will be signified by a line with a # mark as the first character. Output data

You should output, one answer per line, the factorial of each given number, in the format shown in the sample output below. Sample input

```
1
4
3
#
```

Sample output

```
1
24
6
```

Task 2: Change

You are writing a software module for a ticket machine that will be based at a railway station. Your module controls how the ticket machine gives change to customers who pay by cash.

Your module will be provided with a number of data sets. Each data set has two elements: the list of coins available, and the amount of money that needs to be dispensed to the customer as his or her change for the ticket they just bought. The program must decide the set of coins to be dispensed on the basis that:

As few coins as possible must be dispensed.
If there is more than one way to dispense the minimal number of coins, higher denomination coins take precedence.

For each data set there must be exactly one output line. You may assume that if a coin denomination is listed in the coin list, there is sufficient stock of that coin to service the requirement. You may also assume that each denomination will be listed at most once in a data set. Input data

Each transaction your program deals with is represented by a line in the input file. The line begins with a comma-separated list of coin denominations available (in pence), followed by a colon, followed by the amount of money (in pence) that needs to be served to the customer. The coin list may be in any order, i.e. the denominations are not necessarily sorted in any way.

For example, imagine we have coins with values of £1 (100p), 50p, 20p, 10p, 5p, 2p and 1p and we wish to give the customer 57p. The input line would be:

```
100,50,20,10,5,2,1:57
```

The input data may contain space or tab characters, which must be ignored. The end of the input data file will be signified by a line with a # mark as the first character. You may assume that there will be no more than 20 coin denominations, that no line will be more than 100 characters in length, and that there are no malformed lines. Output data

You must list the coins that are to be dispensed. For each coin, state its denomination, followed by 'x', followed by the quantity of that coin that must be dispensed. Coin outputs must be separated with commas. Coins must be listed in descending order of denomination. So the answer for the above example would be:

```
50x1,5x1,2x1
```

... as the most efficient way to dispense 57p from the given set of coins is a single 50p, one 5p and one 2p. There should be no spaces or other whitespace in the output data, and the termination line in the input file (the one beginning with #) should have no corresponding line in the output file. Sample input

```
100,50,20,10,5,2,1:57
100, 10, 50, 20, 5, 2, 1:36
#
```

Sample output

```
50x1,5x1,2x1
20x1,10x1,5x1,1x1
```

(Source IBM Bluemix Challenge, 2015)