Assignment A Confidence interval

When you have a number of noisy observations (represented by a vector x of decimal numbers) a simple confidence interval for the mean is given by the following expression:

$$m \pm 2\frac{s}{\sqrt{n}} \tag{1}$$

where m is the mean, s is the standard deviation, and n is the number of observations. We will use the following definitions:

$$m = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - m)^2}{n-1}}, \tag{2}$$

where x_i are the observations.

Problem definition

Create a function named `confidence` that takes a vector `x` as input and returns the lower and upper confidence interval bounds as a vector `conf` of length two. The first element of the vector must be the lower limit and the second element the upper limit.

Solution template

```
function conf = confidence(x)
% Insert your code
```

Input	
`x`	Observations (vector of decimal numbers).
Output	
`conf`	Lower and upper confidence bounds (vector of length 2).

Example

Consider the following input vector $x = [1,\ 2,\ 4,\ 3,\ 1]$. The mean and standard deviation can be computed as

$$m = \frac{1+2+4+3+1}{5} = 2.2, \qquad s = \sqrt{\frac{(1-2.2)^2 + (2-2.2)^2 + (4-2.2)^2 + (3-2.2)^2 + (1-2.2)^2}{5-1}} = 1.3038,$$

The confidence interval is thus given by

$$2.2 \pm 2 \cdot \frac{1.3038}{\sqrt{5}} \quad = \quad 2.2 \pm 1.1662$$

and the function should thus return the vector $[1.0338,\ 3.3662]$.

A

```
function conf = confidence(x)
    % m = +- 2 s/sqrt(n)
    % m = mean;
    % s = std deviation
    % n = n. of observations
    n = numel(x);
    m = sum(x) / n; % function "mean"!
    s = sqrt((sum((x - m).^2)) / (n -1));% function "std"
    conf = [(m - 2 * s / sqrt(n)), (m + 2 * s / sqrt(n))];
end
```

Assignment B Day of the week

The following formula can be used to compute the day of the week for any date:

$$w = \left(d + C + y + \left\lfloor \frac{y}{4} \right\rfloor\right) \mod 7 \tag{3}$$

Input to the calculation are the date number $d \in \{1 \ldots 31\}$, the month number $m \in \{1 \ldots 12\}$, and the last two digits of the year $y \in \{0 \ldots 99\}$. C is the month code, which can be found from m using the table below. The notation $\lfloor \cdot \rfloor$ means rounding *down* to the nearest smaller integer (the floor function), and mod is the modulo operator (remainder after integer division).

Month (m)	1	2	3	4	5	6	7	8	9	10	11	12
Month code (C)	6	2	2	5	0	3	5	1	4	6	2	4

The result of the computation is the weekday code w which correponds to the following weekday names:

Weekday code (w)	0	1	2	3	4	5	6
Weekday name	`Sun`	`Mon`	`Tue`	`Wed`	`Thu`	`Fri`	`Sat`

Problem definition

Create a function named `weekday` that takes as input the date, month, and year numbers and returns the weekday name as a string written exacly as in the table above.

Solution template

```
function name = weekday(d, m, y)
% Insert your code
```

Input

`d`	Date number (integer, $1 \ldots 31$).
`m`	Month number (integer, $1 \ldots 12$).
`y`	Year number (integer, $0 \ldots 99$).

Output

`name`	Name of the weekday (string).

Example

Consider the date 21 August 2016, which is represented by the input $d = 21$, $m = 8$, $y = 16$. Looking up $m = 8$ in the month code table yields $C = 1$. The weekday code can be computed as:

$$w = \left(21 + 1 + 16 + \left\lfloor \frac{16}{4} \right\rfloor\right) \mod 7 = (21 + 1 + 16 + 4) \mod 7 = 42 \mod 7 = 0 \tag{4}$$

The name of the weekday can then be found by looking up the weekday code in the table, and the string `Sun` is thus the final result.

B

```
function name = weekday(d, m, y)
    % d = 1 ~ 31
    % y = 0 ~ 99 (last 2 digits of the year)
    % C = month code; taken from m and the table
    % m = 1 ~ 12; used as index for C
    C = [6 2 2 5 0 3 5 1 4 6 2 4];
    wn = {'Sun','Mon','Tue','Wed','Thu','Fri','Sat'};
    % Day of the week of any date: w = (d + C + y + abs(y/4)) mod7
    w = mod((d + C(m) + y + floor(y / 4)), 7);

    name = char(wn(w + 1));
    % not necessary the conversion, but I can't know
    % the solution of codejudge, so just to be sure...
end
```

Assignment C Matrix symmetrization

In linear algebra, a symmetric matrix is a square matrix that is equal to its transpose. Given an arbitrary square matrix x, a symmetric matrix y can be constructed as follows:

For *each entry* (i, j) *in the matrix.*
  If $i = j$
    Set $y_{i,j} = x_{i,j}$.
  else
    Set $y_{i,j} = x_{i,j} + x_{j,i}$.
  •
•

Problem definition

Create a function named `symmetrize` that takes a quadratic matrix x as input and returns a symmetrized matrix y computed according to the algorithm above.

Solution template

```
function y = symmetrize(x)
% Insert your code
```

Input	
x	Matrix to be symmetrized (quadratix matrix).
Output	
y	Symmetrized matrix (symmetric matrix).

Example

Consider the following input matrix:

$$x = \begin{bmatrix} 1.2 & 2.3 & 3.4 \\ 4.5 & 5.6 & 6.7 \\ 7.8 & 8.9 & 10.0 \end{bmatrix}.$$

According to the algorithm, the output matrix should then be:

$$y = \begin{bmatrix} 1.2 & 6.8 & 11.2 \\ 6.8 & 5.6 & 15.6 \\ 11.2 & 15.6 & 10.0 \end{bmatrix}.$$

C

```
function y = symmetrize(x)
    % x = quadratic matrix;
    % y = symmetrised matrix;
    N = size(x, 1);

    % check if there's a more efficient way if there's time
%     for i = 1:N
%         for j = 1:N
%             if (i == j)
%                 y(i, j) = x(i, j);
%             else
%                 y(i,j) = x(i, j) + x(j, i);
%             end
%         end
%     end
    y = x + x' - diag(diag(x));
end
```

Assignment D Volume difference

A hypersphere is a generalization of a circle (2-d) and a sphere (3-d) to n-dimensinoal space. The volume of a hypersphere is given by:

$$V_s = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} R^n, \tag{5}$$

where n is the dimension of the space, R is the radius of the hypersphere, and $\Gamma(\cdot)$ is the gamma function which is implemented in Matlab as the function `gamma`.

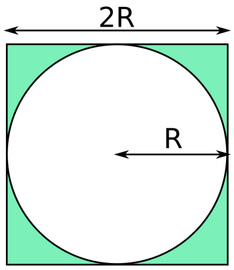
A hypershpere with radius R will fit inside a hyper-cube with side length $2R$. The volume of such a hyper-cube is given by:

$$V_c = (2R)^n. \tag{6}$$

The difference between the two volumes is given by

$$V_d = V_c - V_s, \tag{7}$$

and is illustrated by the colored areas in the figure below, which shows the case in the 2-dimensional setting.



Problem definition

Create a function named `voldif` that takes the radius R and the dimensionality n as input, and returns the difference between the volumes of the hypercube and hypersphere, V_d.

Solution template

```
function Vd = voldif(R, n)
% Insert your code
```

Input	
R	Radius (non-negative decimal number).
n	Dimensionality (positive integer).
Output	
Vd	Difference between volume of hypercube and hypersphere (decimal number).

Example

Consider a radius $R = 5$ and $n = 2$ dimensions. The volumes (which are actually areas in the 2-dimensional case) can be computed as:

$$V_s = \frac{\pi^{\frac{2}{2}}}{\Gamma(\frac{2}{2}+1)} 5^2 \approx 78.54, \qquad V_c = (2 \cdot 5)^2 = 100,$$

and the difference, which is the final result, is given by

$$V_d = 100 - 78.54 = 21.46\,.$$

```
function Vd = voldif(R, n)
    % Inputs:
    % n = dimension of the space
    % R = radius

    Vs = ((pi^(n/2)) / (gamma(n/2 + 1)))*(R^n);
    Vc = (2*R)^n;
    Vd = Vc - Vs;
end
```

Assignment E String comparison

A measure of dis-similarity between strings can for example be used to compare two strings from different data sources. In this exercise we will work with a simple measure defined as the number of different letters `a`–`z` that occur in one and only one of the two strings. If, for example, one string contains two `a`'s and the other contains none, this counts as a difference of 1, and if one string contains two `a`'s and the other contains one `a`, this does not count as a difference since both strings contain the letter `a`. You may assume that the inputs contain only lower case characters `a`–`z`.

Problem definition

Create a function named `stringcompare` that takes as input two strings and returns their dis-similarity as defined above.

Solution template

```
function disSimilarity = stringcompare(string1, string2)
% Insert your code
```

Input	
`string1`, `string2`	Strings to be compared (string).
Output	
`disSimilarity`	Dis-similarity measure (integer).

Example

Consider comparing the two strings `aardvark` and `artwork`. The letters `a`, `r`, and `k` occur in both strings and can be ignored. The two letters `d` and `v` occur only in the first string, and the three letters `t`, `w`, and `o` occur only in the second string. Thus the dis-similarity is $2 + 3 = 5$, and the function must return the number 5.

E

```
function disSimilarity = stringcompare(string1, string2)
    % Assumption: only lower case letters

    % v1 contains the number of occurrences of each letter in string1
    % v2 the same for string2
    % both are vectors of 26 elements (a subfunction can do this task)
    p = 'a';
    q = 'z';
%     for i = 1:26
%         v1(i) = sum(string1 == (96 + i));
%         v2(i) = sum(string2 == (96 + i));
%     end
%     disSimilarity = sum((v1 == 0 & v2 ~= 0) |  (v2 == 0 & v1 ~= 0));

    % Compute string dis-similarity using set xor
    disSimilarity = length(setxor(string1, string2));

end
```