Assignment A Exponential series expansion

The exponential function e^x can be approximated by the following power series:

$$f(x) = \sum_{i=0}^{N-1} \frac{x^i}{i!}$$

where $i!$ denotes the factorial of i, and N is the number of terms.

Problem definition

Create a function named `eseries` that takes as input x and N, and evaluates the above expression.

Solution template

```
function f = eseries(x, N)
% Insert your code
```

Input

`x`	x-value to evaluate the approximation (decimal number).
`N`	Number of terms (positive whole number).

Output

`f`	The approximation of the exponential function at x (decimal number).

Example

Consider evaluating $f(x)$ at $x = 1.23$ with $N = 5$ terms. The terms can be computed as (here shown with five decimals)

$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
1.00000	1.23000	0.75645	0.31014	0.09537

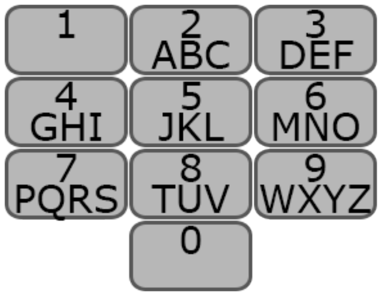
The sum can then computed as 3.39196 which is the final result.

A

```
function f = eseries(x, N)
    % x (decimal number) = point to evaluate
    % N (positive whole number) = number of terms
    v = 0:(N - 1);
    v = x.^v ./ (factorial(round(v)));
    f = sum(v);
end
```

Assignment B Alpha to phone number

On a phone keypad, each letter of the alphabet is assigned to one of the digits 2-9. This makes it possible to write alpha-numeric phone numbers using a mix of letters and digits (by replacing digits in the phone number by the corresponding letters).



Problem definition

Create a function named `alphaToPhone` that takes as input an alpha-numeric (letters and digits) phone number as a string, and returns the corresponding numeric (only digits) phone number as a string. You may assume that all letters in the input are given as upper case.

Solution template

```
function phone = alphaToPhone(alpha)
% Insert your code
```

Input	
`alpha`	Alpha-numeric phone number (string containing letters and digits).
Output	
`phone`	Numeric phone number (string containing only digits).

Example

Consider the alpha-numeric phone number `4525DTU1`. Converted to a numeric phone number, it should be `45253881`.

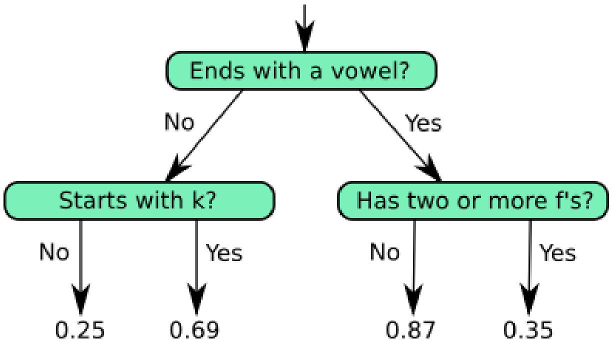
B

```
function phone = alphaToPhone(alpha)
    % alpha (string) = letters and digits
    % phone (string) = only digits

    % numbers unchanged
    phone(alpha <= '9' & alpha >= '0') = alpha(alpha <= '9' & alpha >= '0');
    %letters converted
    phone(alpha >= 'A' & alpha <= 'C') = '2';
    phone(alpha >= 'D' & alpha <= 'F') = '3';
    phone(alpha >= 'G' & alpha <= 'I') = '4';
    phone(alpha >= 'J' & alpha <= 'L') = '5';
    phone(alpha >= 'M' & alpha <= 'O') = '6';
    phone(alpha >= 'P' & alpha <= 'S') = '7';
    phone(alpha >= 'T' & alpha <= 'V') = '8';
    phone(alpha >= 'W' & alpha <= 'Z') = '9';
end
```

Assignment C Guess the gender

You are given a simple "decision tree" below that aims at predicting the gender of a person given his or her name.



The four numbers indicated at the possible outcomes are the probability that the given name is female.

Problem definition

Create a function named `genderGuess` that takes as input a name as a string and computes the probablity that the name is female based on the decision tree above. You may assume that the input name consists only of lower case letters `a`–`z`. As vowels we consider the letters `a`, `e`, `i`, `o`, `u`, and `y`.

Solution template

```
function pFemale = genderGuess(name)
% Insert your code
```

Input	
`name`	Name (string).
Output	
`pFemale`	Probability that name is female (decimal number).

Example

Consider the name `affonso`. We start at the top of the decision tree. Since the name ends with a vowel, `o`, we go down to the right in the decision tree. Next, since the name contains two or more `f`'s, we go down to the right again. The final result, which can be read off by the arrow, is that the probability of a female name is 0.35.

C

```
function pFemale = genderGuess(name)
    % assumptions:
    % 1 - 'name' is lower case;
    % 2 - vowels: a, e, i , o, u, y;
    L = length(name);
    lt = name(L);
    if (lt == 'a' || lt == 'e' || lt == 'i' || lt == 'o' || lt == 'u' || lt == 'y')
        if (sum(name == 'f')) >= 2
            pFemale = 0.35;
        else
            pFemale = 0.87;
        end
    else
        if (name(1) == 'k')
            pFemale = 0.69;
        else
            pFemale = 0.25;
        end
    end
end
```

Assignment D Birthday problem

The socalled "birthday problem" consists of calculating the probability that at two (or more) people within a population of n people have the same birthday. This probability can be computed as:

$$P(n) = 1 - \exp\left(\ln\Gamma(k+1) - \ln\Gamma(k-n+1) - n\log(k)\right)$$

where $\log(\cdot)$ and $\exp(\cdot)$ are the natural logarithm and exponential function, and $\ln\Gamma(\cdot)$ is the socalled log-gamma function which is implemented in Matlab as the function `gammaln`. The number k is the number of days in a year, which we will set to $k = 365$, and we can assume that $2 \geq n \geq k$.

Problem definition

Create a function named `birthday` that takes as input the size of the population, n, and returns the probability $P(n)$ that two (or more) people within the population have the same birthday.

Solution template

```
function P = birthday(n)
% Insert your code
```

Input	
`n`	The number of people in the population (positive whole number).
Output	
`P`	The probability that two (or more) persons have the same birthday (decimal number).

Example

Consider at population of size $n = 23$. The probability can be computed as follows (show with for decimals):

$$\begin{aligned} P(23) &= 1 - \exp\left(\ln\Gamma(365+1) - \ln\Gamma(365-23+1) - 23\log(365)\right) && (1)\\ &= 1 - \exp(1792.3316 - 1657.3419 - 135.6976) && (2)\\ &= 0.5073 && (3) \end{aligned}$$

D

```
function P = birthday(n)
    % Assume k <= n <= 2
    % P(n) = 1 - exp((gammaln(k+1))-gammaln(k-n+1)-n*ln(k))
    % k = 365

    % n (positive whole number) = n. of people in a population
    % P (decimal number)= probability of >= 2 persons have same birthday
    k = 365;
    P = 1 - exp((gammaln(k + 1)) - gammaln(k - n + 1)- n*log(k));
end
```

Assignment E Matrix search

You are given a matrix of whole numbers, where both the rows and the columns are sorted in increasing order. For example, the matrix could look as follows:

$$\begin{bmatrix} 1 & 2 & 6 & 10 \\ 3 & 7 & 7 & 13 \\ 7 & 9 & 11 & 14 \end{bmatrix}.$$

Let us denote the matrix A, its dimensions $M \times N$, and its elements $a_{i,j}$. The following algorithm is an efficient way of finding out if and where a specific number, x, occurs in the matrix:

1. Start at the top right corner of the matrix, $i = 1$, $j = N$.
2. Examine the number $a_{i,j}$:
   (a) If $a_{i,j} = x$ you are done. Return the result $[i, j]$.
   (b) Else, if $a_{i,j} > x$ go one step to the left, $j \leftarrow j - 1$.
   (c) Else, if $a_{i,j} < x$ go one step down, $i \leftarrow i + 1$.
3. If you are within the matrix, i.e. $i \leq M$ and $j > 0$, repeat from 2. Otherwise, return the result $[0, 0]$.

Problem definition

Create a function named `matrixSearch` that takes as input a matrix A as described above and a number x to search for in the matrix. The function must return a vector of the coordinates $[i, j]$ of the first occurence of x as found by the algorithm above. If x does not occur in the matrix, the function must return the vector $[0, 0]$.

Solution template

```
function index = matrixSearch(A, x)
% Insert your code
```

Input	
`A`	Row and column sorted matrix ($M \times N$) with whole numbers.
`x`	Number to search for (whole number).
Output	
`index`	Row and column coordinates of the found number (vector of length 2). Return $[0, 0]$ if the number does not occur in the matrix.

Example

Consider the matrix shown above where $M = 3$ and $N = 4$, and let us look for the number $x = 7$. We start at the top right corner at $a_{1,4} = 10$. Since $10 > x$ we move left to $a_{1,3} = 6$. Since $6 < x$ we move down to $a_{2,3} = 7$. Since this is equal to x, we return the result $[2, 3]$.

E

```
function index = matrixSearch(A, x)
    % A = matrix of whole numbers, MxN
    % Both Cols and Rows sorted in increasing order
    % index = vector 1x2 = coordinates of x in A
    [M, N] = size(A);

    % start: upper right corner (1,N)
    i = 1;
    j = N;
    % if a = x -> return it
    % if a > x -> one coloumn to the left (j = j - 1)
    % if a < x -> one row down (i = i + 1)
    % if i <= M and j > 0 repeat, otherwise return [0,0]
    while ((i <= M) && (j > 0))
        if (A(i,j) == x)
            index = [i, j];
            return
        elseif(A(i,j) > x)
            j = j - 1;
        elseif(A(i,j) < x)
            i = i + 1;
        end
    end
    index = [0, 0];
end
```