Assignment A N-best scores

You are creating recommendation system for a webshop, which must recommend N products to the customer. You are given a list of $K > N$ matching scores (decimal numbers) which indicate how interested the customer is in each of K products.

Problem definition

Create a function named `nBest` that takes as input a vector of length K and an integer N, and returns a vector containing the N largest values from the original vector in the same order as they appear. You may assume that all the values are unique (no duplicates).

Solution template

```
function bestValues = nBest(allValues, N)
% Insert your code
```

Input	
`allValues`	Matching scores (vector of decimal numbers).
`N`	Number of products to recommend (positive integer).

Output	
`bestValues`	N best matching scores (vector of decimal numbers).

Example

Consider the following input matching scores:

$$[12.5,\ 13.6,\ -9.1,\ 17.5,\ 15.3,\ 10.5]$$

If the desired number of recommendations is $N = 3$, the result should be the vector:

$$[13.6,\ 17.5,\ 15.3]$$

A

```
function bestValues = nBest(allValues, N)
    [~, i] = sort(allValues);
    n = numel(allValues);
    bestValues = allValues(sort(i((n - N + 1):n)));
end
```

Assignment B Time dilation

In the theory of relativity, on a fast moving spaceship time will go slower than on earth according to the following formula:

$$T(t, v) = \frac{t}{\sqrt{1 - \frac{v^2}{c^2}}},$$

where T is the time on earth, t is the time on the spaceship, v is the speed of the spaceship (relative to earth), and c is the speed of light. We will use $c = 299\,792\,458$ [meter/second].

Problem definition

Write a function named `timeDilation` that takes as input a vector of N_t values for the time on the spaceship (in seconds) as well as a vector of N_v values of the velocity (in meters/second) and computes a matrix of size $N_t \times N_v$ that contains the corresponding time (in seconds) on earth for each combination of the given times and velocities of the spaceship.

Solution template

```
function T = timeDilation(t, v)
% Insert your code
```

Input	
`t`	Time on spaceship, t (vektor).
`v`	Velocity of spaceship, v (vektor).

Output	
`T`	Time on earth, T (matrix).

Example

Assume that we are given the following times and velocities for the spaceship:

$$t = [5,\ 10.5,\ 16], \qquad v = [1.1 \cdot 10^8,\ 2.2 \cdot 10^8]$$

We thus have $N_t = 3$ and $N_v = 2$, and the resulting matrix should then be:

$$\overline{\overline{T}} = \begin{bmatrix} T(t_1, v_1) & \cdots & T(t_1, v_{N_v}) \\ \vdots & & \vdots \\ T(t_{N_t}, v_1) & \cdots & T(t_{N_t}, v_{N_v}) \end{bmatrix} = \begin{bmatrix} 5.375 & 7.360 \\ 11.287 & 15.457 \\ 17.200 & 23.553 \end{bmatrix}$$

where, for example, element (3,1) is computed as:

$$T(t_3, v_1) = \frac{16}{\sqrt{1 - \frac{(1.1 \cdot 10^8)^2}{299\,792\,458^2}}} \approx 17.200,$$

```
function T = timeDilation(t, v)
    Nt = numel(t);
    c = 299792458;
    for i=1:Nt
        T(i,:) = t(i) ./ sqrt(1 - (v ./ c).^2);
    end
end
```

Assignment D Industrial classification

A company can be classified as one of the following industries.

Code	Industry name
0000–0999	`Agriculture`
1000–1499	`Mining`
1500–1999	`Construction`
2000–3999	`Manufacturing`
4000–4999	`Transportation`
5000–5999	`Trade`
6000–6999	`Finance`
7000–8999	`Services`
9000–9999	`Public`

Problem definition

Create a function named `industry` that takes as input a text string representing an industry code and returns the corresponding industry name as a text string (written exactly as in the table above). The input consists of 4 characters which can be either digits (0–9) or the letter `x`. The `x` works as a "wild card" that can stand for any digit. You can assume that the given code always matches exactly one of the code ranges in the table.

Solution template

```
function industryName = industry(industryCode)
% Insert your code
```

Input
`industryCode`

Output
`industryName`

Example

Consider the input string `35xx`. Since the two `x` characters can stand for any digit, this would represent a number between 3500 and 3599. Since these codes fall within 2000–3999: Manufacturing, the string `Manufacturing` must be returned.

D

```
function industryName = industry(industryCode)
    industryCode(industryCode == 'x') = '0';
    n = str2num(industryCode);
    if(n < 1000)
        industryName = 'Agriculture';
    elseif(n < 1500)
        industryName = 'Mining';
    elseif(n < 2000)
        industryName = 'Construction';
    elseif(n < 4000)
        industryName = 'Manufacturing';
    elseif(n < 5000)
        industryName = 'Transportation';
    elseif(n < 6000)
        industryName = 'Trade';
    elseif(n < 7000)
        industryName = 'Finance';
    elseif(n < 9000)
        industryName = 'Services';
    elseif(n < 10000)
        industryName = 'Public';
    end
end
```

Assignment E Task dispatch

You are working on a production system in which a number of tasks are to be processed on a number of processing units. Each task requires a certain number of resources, and each processing unit has a certain number of resources available. You must create a task dispatching system which will assigne each task to a processing, taking the number of resources into account. To assign the tasks you must use the following algorithm:

Let $t(n)$ denote the number of resources required for task n.
Let $u(k)$ denote the number of resources available on processing unit k.
Let $a(n)$ denote the number of the processing unit to which task n is assigned.
Loop *over all processing units (let k denote the processing unit).*
  Loop *over all tasks that have not yet been assigned (let n denote the task).*
    If *there is enough room on processing unit k for task n, i.e. if $u(k) \geq t(n)$.*
      Assign task n to processing unit k: $a(n) \leftarrow k$.
      Update the number of resources on processing unit k: $u(k) \leftarrow u(k) - t(n)$.

Problem definition

Create a function named `taskDispatch` that takes as input a vector containing the number of required resources for N tasks as well as a vector containing the number of available resources for K processing units. The function must return a vector of length N containing the number of the processing unit (number between 1 and K) to which each task was assigned according to the algorithm above. Tasks that have not been assigned to any processing unit (because of lack of resources) should be marked by zero.

Solution template

```
function assignment = taskDispatch(tasks, units)
% Insert your code
```

Input	
`tasks`	The number of resources required for each task (vector).
`units`	The total number of available resource for each processing unit (vector).

Output	
`assignment`	Assignment (processing unit number) for each task (vector).

Example

Consider the situation where we have 6 tasks and 2 processing units with the following number of resources required/available:

$$t = [5,\ 7,\ 4,\ 2,\ 3,\ 5], \quad u = [11,\ 13].$$

The first processing unit has $u(1) = 11$ resources. We assign to it the first task and update $u(1) = 11-5 = 6$. There is not room the second task, so we move on and assign the third task and update $u(1) = 6-4 = 2$. There is room for the fourth task also, so we assign that and update $u(1) = 2-2 = 0$. There is not room for any of the following tasks, so we move on to the second proceesing unit. The first task is already assigned, so we assign the second task and update $u(2) = 13-7 = 6$. The third and fourth task are already assigned, so we move on and assign the fifth task and update $u(2) = 6-3 = 3$. There is not room for the sixth task, so that is not assigned to any processing unit, and the algorithm is finished. The final assignment should thus be:

$$a = [1,\ 2,\ 1,\ 1,\ 2,\ 0].$$

E

```
function assignment = taskDispatch(tasks, units)
    n = numel(tasks);
    k = numel(units);

    assignment = zeros(1, n);
    for k = 1:k
        for n = 1:n
            if (assignment(n) == 0)
                if units(k) >= tasks(n)
                    assignment(n) = k;
                    units(k) = units(k) - tasks(n);
                end
            end
        end
    end
end
```