Assignment A Normal weight

The body mass index (BMI) is defined as

$$\texttt{BMI} = \frac{w}{h^2}$$

where w is body weight (mass) in kilograms, and h is the height in metres. A person is categorized as being of normal weight if his BMI falls in range from 18.5 to 25 $\mathrm{kg\,m^{-2}}$, both endpoints included.

Problem definition

Write a function `normalWeight` that takes height in meters as input. The function should return a vector representing the normal weight range, given as the smallest and the largest weight in whole kilograms, which when converted to BMI fall into normal range.

Solution template

```
function w = normalWeight(h)
% Insert your code
```

Input	
`h`	Height in meters (decimal number).
Output	
`w`	Interval for normal weight in kilograms (vector with 2 elements).

Example

Say a height is given as `h` = 1.73. The weight limits of the normal range can be calculated as

$$w_{\text{lower}} = 18.5 \cdot 1.73^2 = 55.3687 \quad \text{and} \quad w_{\text{upper}} = 25 \cdot 1.73^2 = 74.8225\,. \qquad (1)$$

We need weight in whole kilograms, so numbers should be rounded. To ensure that weight falls into the normal range, the lower limit should be rounded up and the upper limit should be rounded down. The final output of the function is the vector $[56, 74]$.

A

```
function weight = normalWeight(h)
    % BMI = w / h^2
    % normal weight = 18.5 ~ 25 kg m^-2
    % Interval: 18.5*h ~ 25*h
    weight = [18.5 25]*(h^2);
    weight(1) = ceil(weight(1));
    weight(2) = floor(weight(2));
    w = weight;
end
```

Assignment B Dance class

A dance class is considered valid if all of the following conditions are met:

- there is no less than 10 participants,
- there is no more than 30 participants,
- when the participants are in dance couples (one male and one female), there is no more than 3 unpaired (only male or only female) participants.

For registration purposes a vector is used with number 0 corresponding to the male and 1 to female participant.

Problem definition

Write a function named `danceClass` that takes a vector corresponding to registered participants as input and returns a string `valid` or `invalid` indicating whether the dance class is valid or not.

Solution template

```
function s = danceClass(v)
% Insert your code
```

Input	
`v`	Class registration (vector containing zeros and ones).
Output	
`s`	Class validity (string).

Example

Say we have following vector as input

$$[\,1\quad 1\quad 1\quad 0\quad 0\quad 1\quad 0\quad 1\quad 1\quad 1\quad 0\,].$$

There are 11 registered participants (elements in the vector). Number 11 is neither smaller than 10, nor larger than 30, so the first two conditions are met. There are 7 female participants (ones in the vector) and 4 male participants (zeros in the vector). This makes 4 male-female couples, and leaves 3 unpaired female participants. Number 3 is not larger than 3, so last condition is also met. The function should return `valid`.

B

```
function s = danceClass(v)
    % Valid Class:
    % 1 - n. participants >= 10 and <= 30;
    % 2 - No more than 3 unpaired participants
    % v (vector): 0 = male; 1 = female
    n = length(v);
    if  ((n <= 30) && (n >= 10))
        males_cnt = sum(v == 0);
        females_cnt = sum(v == 1);
        if (abs(males_cnt - females_cnt) > 3)
            s = 'invalid';
        else
            s = 'valid';
        end
    else
        s = 'invalid';
    end
end
```

Assignment C Survival temperature

To survive in cold temperatures, humans must either maintain a sufficiently high metabolic rate, or regulate heat loss by covering their body with clothes made of insulating material. The expression giving the lowest temperature for survival T in degrees Celsius as a function of metabolic heat production M in $\mathrm{W\,m^{-2}}$ and a thermal conductance (for clothes and tissue) g in $\mathrm{mol\,m^{-2}\,s^{-1}}$ is

$$T = 36 - \frac{(0.9M - 12)(g + 0.95)}{27.8g}\,. \tag{2}$$

The reasonable range for metabolic heat production is from 50 $\mathrm{W\,m^{-2}}$ for sleep to 500 $\mathrm{W\,m^{-2}}$ for heavy activity, e.g. sports. The reasonable range for thermal conductance is from 0.04 $\mathrm{mol\,m^{-2}\,s^{-1}}$ for winter sleeping bag to 0.45 $\mathrm{mol\,m^{-2}\,s^{-1}}$ for no clothing.

Problem definition

Write a function named `survivalTemperature` that takes a vector of values for the metabolic heat production `M` and a vector of values of the thermal conductance `g` as input. The function should return a matrix `T` of size $N_M \times N_g$, where N_M is the number of elements of `M` and N_g is number of elements in `g`. Elements of `T` should be the lowest survival temperature for the corresponding values from `M` and `g`. If some of the input values from `M` and `g` fall outside the reasonable range for that input, the function should not return a matrix, but instead a string `RangeError`.

Solution template

```
function T = survivalTemperature(M,g)
% Insert your code
```

Input	
`M`	Values for metabolic heat production M (vector with N_M elements).
`g`	Values for thermal conductance g (vector with N_g elements).
Output	
`T`	Values for survival temperature T ($N_M \times N_g$ matrix) or an error string.

Example

Consider following input

$$\mathtt{M} = [\ 50 \quad 200 \quad 300\ ]\,, \qquad \mathtt{g} = [\ 0.20 \quad 0.14\ ]\,. \tag{3}$$

We have $N_M = 3$, $N_g = 2$, and the resulting 3×2 matrix should be

$$\mathtt{T} = \begin{bmatrix} 29.1745 & 26.7580 \\ 1.2518 & -11.0504 \\ -17.3633 & -36.2559 \end{bmatrix}, \tag{4}$$

where, for example, element (2,1) of matrix `T` is computed as

$$T_{(2,1)} = 36 - \frac{(0.9 \cdot 200 - 12)(0.20 + 0.95)}{27.8 \cdot 0.20} = 1.2518\,. \tag{5}$$

```matlab
function T = survivalTemperature(M,g)
    % Range M: 50  ~ 500
    % range g: 0.04 ~ 0.45
    % Error if one of the element of M or g fall outside the range
    if (any(M < 50 | M > 500) || any(g < 0.04 | g > 0.45))
        T = 'RangeError';
    else
        % Lowest T survival = 36 - (0.9M-12)(g+0.95)/(27.8g)
        % T(i,j) = lowest survival T for corresponding M and g
        NM = length(M);
        Ng = length(g);
        for (i = 1:NM)
            T(i, 1:Ng) = 36 - (M(i)*0.9 - 12)*(g(1:Ng)+0.95) ./ (27.8*g(1:Ng));
        end
    end
end
```

Assignment D Green eyes

The following chart gives probabilities (in percents) for child eye colors based on the eye colors of the parents.

Parents		Child ● (brown)	● (blue)	● (green)
● + ●	(brown + brown)	75	6	19
● + ●	(brown + blue)	50	50	0
● + ●	(brown + green)	50	12	38
● + ●	(blue + blue)	0	99	1
● + ●	(blue + green)	0	50	50
● + ●	(green + green)	0	25	75

Problem definition

Write a function `greenEyes` that takes two strings representing eye colors of the parents (in any order) as input. The strings are expected to be written in lower case as in the table above. The function should return the probability (in percents) of the child eye color being green.

Solution template

```
function P = greenEyes(p1,p2)
% Insert your code
```

Input	
p1	Eye color of one parent (string).
p2	Eye color of second parent (string).
Output	
P	The probability in percent of the child eye color being green (number).

Example

Consider following input

$$p1 = \texttt{green}\,, \quad p2 = \texttt{brown}\,. \tag{6}$$

According to the table, the green and brown combination (brown+green) gives the 38 percent probability for a child having green eyes. Thus, the number 38 should be returned.

D

```
function P = greenEyes(p1,p2)
    % p1, p2 = eye colours of parents (any order, lower case)
    % P = probability in % of child's colour being GREEN
    G = [19 0 38 1 50 75];
    if (strcmp(p1, 'brown') && strcmp(p2, 'brown'))
        P = G(1);
    elseif ((strcmp(p1, 'brown') && strcmp(p2, 'blue')) || (strcmp(p2, 'brown') &&↙
strcmp(p1, 'blue')))
        P = G(2);
    elseif ((strcmp(p1, 'brown') && strcmp(p2, 'green')) || (strcmp(p2, 'brown') &&↙
strcmp(p1, 'green')))
        P = G(3);
    elseif ((strcmp(p1, 'blue') && strcmp(p2, 'blue')) || (strcmp(p2, 'blue') &&↙
strcmp(p1, 'blue')))
        P = G(4);
    elseif ((strcmp(p1, 'blue') && strcmp(p2, 'green')) || (strcmp(p2, 'blue') &&↙
strcmp(p1, 'green')))
        P = G(5);
    elseif ((strcmp(p1, 'green') && strcmp(p2, 'green')) || (strcmp(p2, 'green') &&↙
strcmp(p1, 'green')))
        P = G(6);
    end

    P = round(P);
end
```

Assignment E Best buy

A shopper is given a sum of money and a shopping list. He will start at the top of the list, and will be buying items one by one as long as he has enough money to buy the next item. When he does not have enough money to buy the next item, he will go home.

Problem definition

Write a function `bestBuy` which takes a vector containing prices of the items at the shopping list and a number representing the som of money the shopper is given as input. The function should return a total number of items bought by the shopper, before he returned home.

Solution template

```
function n = bestBuy(p,m)
% Insert your code
```

Input	
p	Prices of items (vector of positive numbers).
m	Available money (positive number).
Output	
n	Number of bought items (whole number).

Example

Say we have a following input

$$\mathtt{p} = [\ 5 \quad 4 \quad 6 \quad 2 \quad 9 \quad 1 \quad 1 \quad 4\ ], \quad \mathtt{m} = 16\,. \tag{7}$$

The spending looks like this

number of bought items	0	1	2	3
money left	16	11	7	1
price of next item	5	4	6	2

After 3 bought items the shopper has spent $5 + 4 + 6 = 15$ money and has $16 - 15 = 1$ money left, which is not enough to buy the next item on the list which costs 2 money. He returns home with 3 items, and function should return 3.

```
function n = bestBuy(p,m)
    % P (vector) = prices of items
    % m (positive number) = available money
    % n = n. bought items (whole number)
    n = 0;
    L = length(p);
    while (((n+1) <= L) && p(n+1) <= m)
            m = m - p(n+1);
            n = n + 1;
    end
end
```