Assignment B Load balancing

You have a number of different workloads (computer programs) that you want to run on two computers. For each workload you know the expected run-time, and now you wish to distribute the workloads as evenly as possible between the two computers. Say the number of workloads is N you decide to run the first k workloads on the first computer and the last $N - k$ workloads on the second computer.

Problem definition

Create a function named `loadBalance` that takes as input a vector containing the expected runtime of N workloads ($N \geq 2$). The function must return the number k that splits the list of workloads as evenly as possible, i.e. such that the absolute value of the difference between the total expected runtime of the first k and the last $N - k$ workloads is as small as possible. If more than one split results in the same lowest absolute difference you must return the lowest value of k.

Solution template

```
function k = loadBalance(runtime)
% Insert your code
```

Input	
`runtime`	Runtime of N workloads (vector)

Output	
`k`	Number of workloads to run on computer 1 (whole number)

Example

Consider the following vector of runtimes: [5, 2.5, 17, 1.5, 22, 3.5]. We can now compute the sum of the runtime on the two computers and the absolute difference for different values of k:

k	Computer 1	Computer 2	Absolute difference
1	5	$2.5 + 17 + 1.5 + 22 + 3.5 = 46.5$	$\lvert 46.5 - 5 \rvert = 41.5$
2	$5 + 2.5 = 7.5$	$17 + 1.5 + 22 + 3.5 = 44$	$\lvert 44 - 7.5 \rvert = 36.5$
3	$5 + 2.5 + 17 = 24.5$	$1.5 + 22 + 3.5 = 27$	$\lvert 27 - 24.5 \rvert = 2.5$
4	$5 + 2.5 + 17 + 1.5 = 26$	$22 + 3.5 = 25.5$	$\lvert 25.5 - 26 \rvert = 0.5$
5	$5 + 2.5 + 17 + 1.5 + 22 = 48$	3.5	$\lvert 3.5 - 48 \rvert = 44.5$

From this we see that $k = 4$ yields the best load balance (minimum absolute difference), and the function should return the value 4.

B

```
function k = loadBalance(runtime)
    % input (vector) = N expected runtimes >= 2
    % output (number) = number that splits N as evenly as possible
    % abs of difference between tot runtime of first k and the last N-k
    % runtimes is minimum; if more values of k give the same result, the
    % smallest
    n = numel(runtime);
    for i = 1:(n - 1)
        diff(i) = abs(sum(runtime(1:i)) - sum(runtime((i + 1):n)));
    end
    k = find(diff == min(diff), 1);


end
```

Assignment C Nearest color

A color can be represented by three numbers, r, g, and b, which correspond to the amount of red, green, and blue light. In this exercise, it is assumed that the numbers are given as percentages, i.e. between 0 and 100. The following table lists the names and rgb-values of some different colors:

	White	Grey	Black	Red	Maroon	Yellow	Olive	Lime	Green	Aqua	Teal	Blue	Navy	Fuchsia	Purple
r	100	50	0	100	50	100	50	0	0	0	0	0	0	100	50
g	100	50	0	0	0	100	50	100	50	100	50	0	0	0	0
b	100	50	0	0	0	0	0	0	0	100	50	100	50	100	50

We will define the distance between two colors (r_1, g_1, b_1) and (r_2, g_2, b_2) by the maximum absolute difference:

$$D = \max\Big(|r_2 - r_1|,\ |g_2 - g_1|,\ |b_2 - b_1|\Big) \tag{1}$$

Problem definition

Create a function named `nearestColor` that takes a value of r, g, and b as input and returns the name of the nearest color as a string (written exactly as in the table above.) The nearest color is defined as the color in the table to which the distance is smallest. If two or more colors in the table have the same smallest distance, the name of the color which occurs first in the table must be returned.

Solution template

```
function colorName = nearestColor(r, g, b)
% Insert your code
```

Input

`r`, `g`, `b`	Value of r, g, and b (decimal number)

Output

`colorName`	Name of nearest color (string).

Example

If the input is given as $r = 75$, $g = 0$, and $b = 0$, the distance to each of the 16 colors in the table can be computed as:

	White	Grey	Black	Red	Maroon	Yellow	Olive	Lime	Green	Aqua	Teal	Blue	Navy	Fuchsia	Purple
D	100	50	75	25	25	100	50	100	75	100	75	100	75	100	50

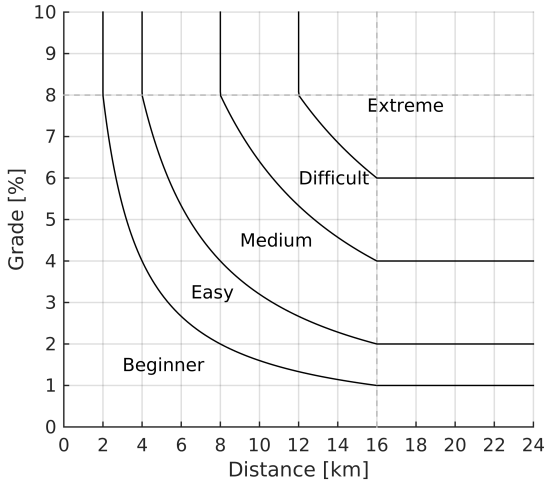
Since the smallest distance is to the colors Red and Maroon, and since Red occurs first in the table, the function must return the string `Red`.

C

```
function colorName = nearestColor(r, g, b)
    % output = name of nearest colour
    % Nearest Colour = colour in the table to which distance is smallest;
    % same smallest distance -> first colour returned
    % distance between 2 colours: D = max(abs diff between r g and b)

    % 15 colours
    Colours = {'White', 'Grey', 'Black', 'Red', 'Maroon', 'Yellow',...
               'Olive','Lime', 'Green', 'Aqua', 'Teal', 'Blue', 'Navy',...
               'Fuchsia', 'Purple'};

    % Coloumns of RGB are triplets r (1st row), g (2nd), b (3rd)
    RGB = [100 50 0 100 50 100 50 0 0 0 0 0 0 100 50;..
           100 50 0 0 0 100 50 100 50 100 50 0 0 0 0;..
           100 50 0 0 0 0 0 0 0 100 50 100 50 100 50];
    D = max([abs(RGB(1,:) - r); abs(RGB(2,:) - g); abs(RGB(3,:) - b)]);

    colorName = Colours{find(D == min(D), 1)};

%     % Define color names
% colorNames = {'White', 'Grey', 'Black', 'Red', 'Maroon', 'Yellow',...
%               'Olive', 'Lime', 'Green', 'Aqua', 'Teal', 'Blue','Navy',...
%               'Fuchsia', 'Purple'};
%
% % Define color rgb-values
% R = [100, 50, 0, 100, 50, 100, 50, 0, 0, 0, 0, 0, 0, 100, 50];
% G = [100, 50, 0, 0, 0, 100, 50, 100, 50, 100, 50, 0, 0, 0, 0];
% B = [100, 50, 0, 0, 0, 0, 0, 0, 0, 100, 50, 100, 50, 100, 50];
%
% % Compute distance to each color
% D = max([abs(R-r); abs(G-g); abs(B-b)]);
%
% % Find nearest color
% [~, colorIndex] = min(D);
%
% % Get color name
% colorName = colorNames{colorIndex};
end
```

Assignment D Mountain climb categorization

A mountain climb can be categorized as Beginner, Easy, Medium, Difficult, or Extreme based on the distance, D, (in kilometers) and the grade, G, (average incline in percent) according to the following figure and table.



	`Beginner`	`Easy`	`Medium`	`Difficult`	`Extreme`
If $G > 8$:	$D < 2$	$2 \leq D < 4$	$4 \leq D < 8$	$8 \leq D < 12$	$12 \leq D$
If $D > 16$:	$G < 1$	$1 \leq G < 2$	$2 \leq G < 4$	$4 \leq G < 6$	$6 \leq G$
Otherwise	$DG < 16$	$16 \leq DG < 32$	$32 \leq DG < 64$	$64 \leq DG < 96$	$96 \leq DG$

Problem definition

Create a function named `climbCategorization` that takes the distance and grade as input and returns the categorization of the climb as a string (written exactly as above).

Solution template

```
function categoryName = climbCategorization(distance, grade)
% Insert your code
```

Input	
`distance`	Distance in kilometers (decimal number)
`grade`	Grade in percent (decimal number)

Output	
`categoryName`	Name of categorization of mountain climb (string).

Example

If the distance and the grade are given by $D = 8$ and $G = 6$ neither of the conditions $G > 8$ or $D > 16$ are fulfilled. We then compute $DG = 6 \cdot 8 = 48$ and since $32 \leq DG < 64$, the string `Medium` must be returned.

D

```
function categoryName = climbCategorization(distance, grade)
    if ( distance<2 || grade<1 || distance*grade<16 )
        categoryName = 'Beginner';
    elseif ( distance<4 || grade<2 || distance*grade<32 )
        categoryName = 'Easy';
    elseif ( distance<8 || grade<4 || distance*grade<64 )
        categoryName = 'Medium';
    elseif ( distance<12 || grade<6 || distance*grade<96 )
        categoryName = 'Difficult';
    else
        categoryName = 'Extreme';
    end
end
```

Assignment E Roman numerals

In the roman numeral system, a number can be written as a sequence of symbols (letters) corrsponding to different values.

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Problem definition

Create a function named `romanToValue` that takes a roman numeral (string) as input and computes its value. The function must use the following algorithm:

1. Convert each symbol into the value it represents.
2. Keep a running total, and a record of the maximum symbol value encountered so far (both initialized to zero). Run through the symbols one by one starting from the right:
   - If the symbol value is greater than or equal to the maximum, add it to the running total and update the maximum.
   - If the symbol value is less than the maximum, subtract it from the running total.

Solution template

```
function value = romanToValue(roman)
% Insert your code
```

Input	
`roman`	Roman numeral (string)

Output	
`value`	Numeric value of the roman numeral (whole number).

Example

Consider the following input: `XCIV`. The symbol values are: 10, 100, 1, 5.

To begin with, the maximum value and the total is set to zero. Starting from the right, the value of the first symbol is 5, which is added to the running total which is now 5 and the new maximum is 5. Next is 1, whis is subtracted from the total which is now 4. Next is 100 which is added to the total which is now 104 and the new maximum is 100. Last is 10 which is subtracted from the total which is now 94, which is the final result.

E

```
function value = romanToValue(roman)
    n = numel(roman);

    num(roman == 'I') = 1;
    num(roman == 'V') = 5;
    num(roman == 'X') = 10;
    num(roman == 'L') = 50;
    num(roman == 'C') = 100;
    num(roman == 'D') = 500;
    num(roman == 'M') = 1000;

    if (n == 1)
        value = num(n);
    else
        mx = 0;
        value = 0;
        for i = n:-1:1
            if (num(i) >= mx)
                value = value + num(i);
                mx = num(i);
            else
                value = value - num(i);
            end
        end
    end
end
```