Attention:

Make sure that you save all your scripts in the same folder, and do not lose your files after the seminar!

Do not copy-paste code! Write it by yourself and try to understand it. You are strongly encouraged to make experiments and to try your own approaches.

Part 1

Our first task is to understand for loops.

1) Launch Matlab.

2) Create a new script. Play a little bit with **for** loops. Try to run:

```
v1 = 1:1:10;

v2 = 1:0.5:5;

v3 = [10, 7, 1, 3, 4];

v4 = 'Algorithms';

for element=v1

   disp(element); %disp function prints element value to output

end
```

Try to change **v1** in **for** loop to **v2**, **v3**, **v4**. You can see in output that each iteration **for** loop picks next element from a vector and assigns its value to the variable **element**. In case of **v4** it picks new letter each iteration (because Matlab treats string as a vector of characters). Note that **disp()** function prints input value on new line each time.

3) Try to write a program that calculates sum of all numbers in **v**:

```
v = [1, 7, 52, 10, -5, -10, -7];

sum = 0;

for element=v

  % you code here

end

disp(sum);
```

4) Modify program (3) to calculate product of all numbers in **v**.

5) Modify program (3) to calculate sum of all **negative** numbers in **v**.

Part 2

In part 2 we will solve the question 1 from the homework for the routes that involve numbers instead of letters.

1) Create a new script. Write **clear;** on the first line.

2) Create a new vector named **route**. Entries of this vector must be positive integer numbers in range 1 to 10 (you can write numbers with your hands, no need to use **rand()**). This vector will be your test data. Make sure that each vector has at least 20 entries.

route = [3, 7, 4, ...]

(use your own numbers, do not use "…")

3) Using **max()** function, find the maximum number in the **route** vector into variable named **number_of_unique_elements**

4) Create a new row vector called **histogram** with size **1** x **number_of_unique_elements**, using **zeros()** function. Remember that if you want to create a vector you must pass two number as an input for the **zeros()** function: number of rows and number of columns. If you pass only one number **n**, the function will create square matrix **n**x**n**.

In this vector we will store number of occurrences of each number in the **route**.

5) Using a **for** loop write a code to iterate over all elements in the **route** vector

```
for facility=route


end
```

You can read it as "for each **facility** in **route**" (but not write!).

6) Increase value of the entry with index = **facility** in the **histogram** vector by 1.

histogram(facility) = …

At this moment your script should look like:

```
clear;

route = [ 1, 2, 3, 4, 4, 4, 5, 6, 7, 8, 9 ];

histogram = zeros(1, max(route));


for facility=route
```

```
    histogram(facility) = histogram(facility) + 1;
end
```

7) Try to run your script now. As a result, you will get the **histogram** of the **route** vector – the number of occurrences of each element of the vector. Check that everything is correct.

Part 3

Now we want to modify our program to work with letters instead of numbers.

1) In the task our input is a string of characters. Thus, at first we want to convert our string of characters to an array of numbers. We will modify our script. Create new empty vector, name it **numeric_route**. In this vector we will store the route in numeric form.

```
numeric_route = zeros(size(route));
```

2) Create new empty vector, call it **letters**. The idea is that we will create mapping of letters in **route** vector to integer numbers. This **letters** vector will contain only unique letters from **route** (i.e. no letter repeats).

3) Create a new **for** loop

```
for facility_index=1:length(route)
        % your code here
end
```

In this loop we will iterate through all indices of the route vector.

1:length(route) creates a vector of integer numbers from **1** to **length(route)** with step 1

For example, **1:5** creates vector **[1, 2, 3, 4, 5]**.

4) Unlike the previous loop, at each iteration our **facility_index** contains the index of an element, not the element value. So, at first line inside the loop we want to get an actual value of the element

```
facility = route(facility_index);
```

5) After we get the value, we want to check if we already have it in our **letters** vector.

At first we create boolean variable with predefined value false. It means that by default we think that letter is not in the **letters** vector, unless we change it to **true**.

```
is_facility_already_in_letters = false;
```

Then we create **for** loop inside **for** loop. In this loop we iterate through **letters** entries in order to check if our current **facility** already in it.

for letter=letters % read as "for each letter in letters"

if letter == facility % if facility from the outer loop equals to the letter in the inner loop

is_facility_already_in_letters = true;

end

end

Then, if facility indeed is not in letters vector, then we add it.

if ~is_facility_already_in_letters %read it as "if not is_facility_already_in_letters"

letters = [letters, facility]; % add element to the vector

end

Now, when we ensure that our **letters** vector contains current facility, we can use index of the facility from the **letters** vector. It will be our mapping of letter to an integer number.

We use **find()** function in order to find indices of elements in **letters** which are equal to **facility** (or simply we looking for current facility letter index in **letters**).

facility_number = find(letters == facility);

And assign this number to the corresponding element of the original route vector

numeric_route(facility_index) = facility_number;

(Note: If you want, you can use **for** loop instead of **find()** function. Or if you want, you can use **find()** function instead of previous **for** loop.)

6) Change your **route** variable to some string of characters. As a result, you should have

route = 'AVVVC';

numeric_route = zeros(size(route));

letters = [];

for facility_index=1:length(route)

facility = route(facility_index);

is_letter_already_in_letters = false;

for letter=letters

if letter == facility

```
        is_letter_already_in_letters = true;
      end
    end

    if ~is_letter_already_in_letters
      letters = [letters, facility];
    end

    facility_number = find(letters == facility);
    numeric_route(facility_index) = facility_number;
end

histogram = zeros(1, max(numeric_route));

for facility=numeric_route
    histogram(facility) = histogram(facility) + 1;
end
```

7) Now we create a function and move our code to it. Create new script named **facilities_histogram.m** (file name and function name must be identical!)

Start the script with the function declaration:

```
function [letters,histogram]=facilities_histogram(route)
```

% your code here

```
end
```

8) Put your code (except **route = ...** and **clear;** lines) to the function, between function declaration and **end** keyword.

9) Now you can call your function with an arbitrary input, using the following syntax:

```
facilities_histogram(route)
```

For example,

```
current_route = 'ABBABABBABAB';
```

new_route = 'BABABABABCBBCBAB';

[letters1, histogram1] = facilities_histogram(current_route);

[letters2, histogram2] = facilities_histogram(new_route);

10) Write by yourself code to check if route satisfies the conditions from the task using the **facilities_histogram** function. Remember that **letters1** and **letters2** can have letters in different order. Make sure that your code works for any arbitrary input (any number of different letters in any order). Test it with different instances.

Notes:

1) The code above can be simplified by using hashmaps (https://en.wikipedia.org/wiki/Hash_table). They are very similar to vectors, but instead of number we can use characters or strings as indices. One can find more information about hash maps in the Matlab documentation (http://se.mathworks.com/help/matlab/ref/containers.map-class.html).