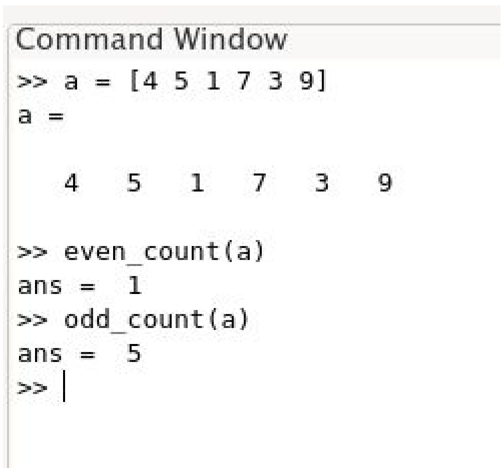
# Software Tools Lab 8

Roll No.: U18CO021

**1. Write a function that takes a vector and counts how many even numbers there are. [Also odd numbers]**

```
% even count
function count = even_count(arr)
  count = 0;
  for i = 1:length(arr)
    if ~rem(arr(i), 2)
      count += 1;
    endif
  endfor
endfunction

% odd count
function count = odd_count(arr)
  count = 0;
  for i = 1:length(arr)
    if rem(arr(i), 2)
      count += 1;
    endif
  endfor
endfunction
```

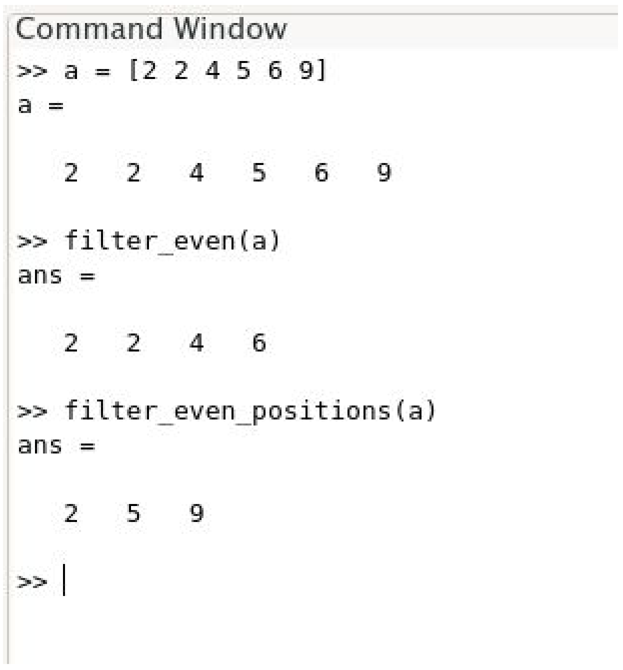


```
Command Window
>> a = [4 5 1 7 3 9]
a =

   4   5   1   7   3   9

>> even_count(a)
ans =  1
>> odd_count(a)
ans =  5
>> 
```

**(i) Write a function that returns a vector of all the even valued elements in a given vector. E.g., if the given vector is [2 2 4 5 6 9], the returned vector is [2 2 4 6]**

**(ii) Write a function that returns a vector of all the elements in even positions in a given vector, i.e. every second element in the vector. E.g., if the given vector is [2 2 4 5 6 9], the returned vector is [2 5 9].**

```
% 1
function even_vector = filter_even(arr)
  even_vector = [];
  for i = 1:length(arr)
    if ~rem(arr(i), 2)
      even_vector = [even_vector arr(i)];
    end
  end
end

% ii
function even_vector = filter_even_positions(arr)
  even_vector = [];
  for i = 1:length(arr)
    if ~rem(i, 2)
      even_vector = [even_vector arr(i)];
    end
  end
end
```

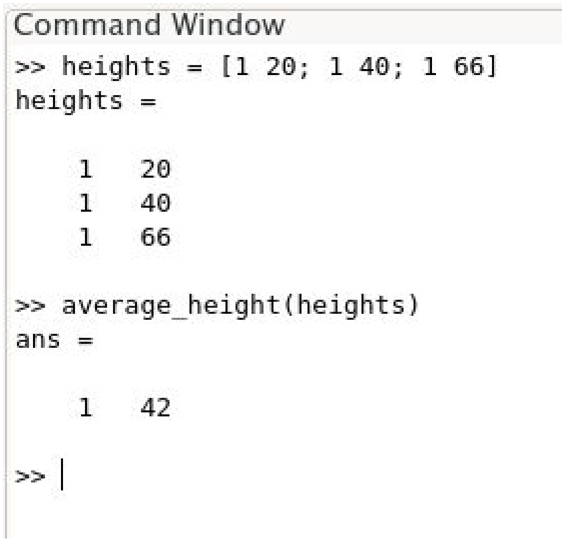


```
Command Window
>> a = [2 2 4 5 6 9]
a =

   2   2   4   5   6   9

>> filter_even(a)
ans =

   2   2   4   6

>> filter_even_positions(a)
ans =

   2   5   9

>> |
```

**2. Given a vector of heights of students in a class (in meters), return the average height.**

**Given an n by 2 that has the heights of students in a class, where the first column is the part of the height before the decimal point (in meters), and the second column is the part of the height after the decimal point (i.e. in cm), return an n by 1 vector that contains the height of the students in m. Also: in cm. E.g., a row [1 63] means that the height of the person is 1m and 63cm.**

```
function avg = average_height(heights)
  s = 0;
  for i = 1:length(heights)
    s = s + height_to_cm([heights(i, 1) heights(i, 2)]);
  end
  mean = s / length(heights);
  avg = cm_to_height(mean);
end

function cm = height_to_cm(height)
  cm = 100 * height(1) + height(2);
end

function height = cm_to_height(cm)
  height = [floor(cm/100), rem(cm, 100)];
end
```

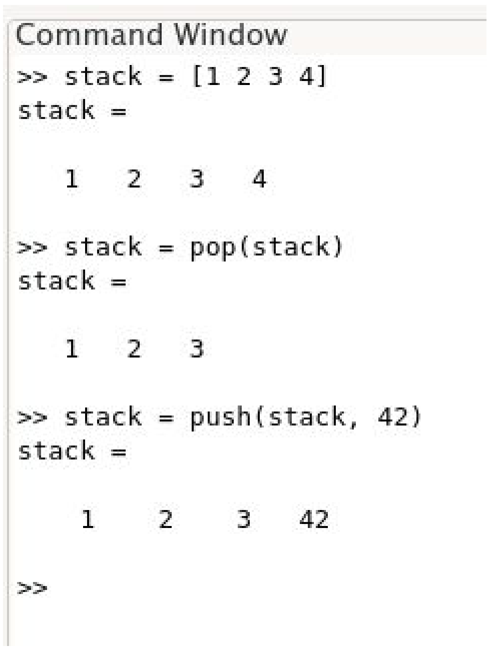


```
Command Window
>> heights = [1 20; 1 40; 1 66]
heights =

    1   20
    1   40
    1   66

>> average_height(heights)
ans =

    1   42

>> |
```

### 3. Push and pop from a stack

```
function stack = pop(stack)
    if length(stack)
        stack = stack(1:end - 1);
    end
end
```

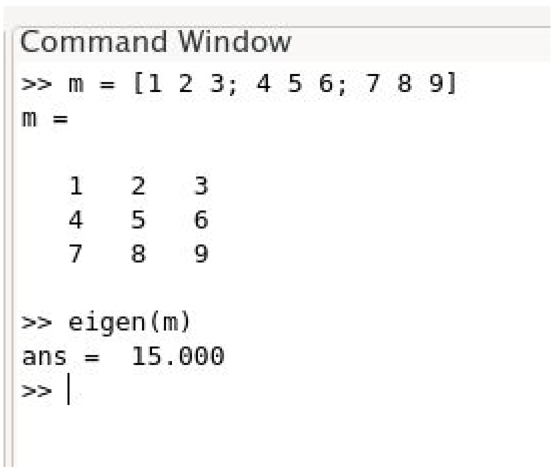
```
function stack = push(stack, n)
  stack = [stack n];
end
```



```
Command Window
>> stack = [1 2 3 4]
stack =

   1   2   3   4

>> stack = pop(stack)
stack =

   1   2   3

>> stack = push(stack, 42)
stack =

    1    2    3   42

>>
```

**4. Write a function that takes a matrix and returns the sum of its Eigenvalues**

```
function s = eigen(sqr_matrix)
    s = sum(eig(sqr_matrix));
end
```

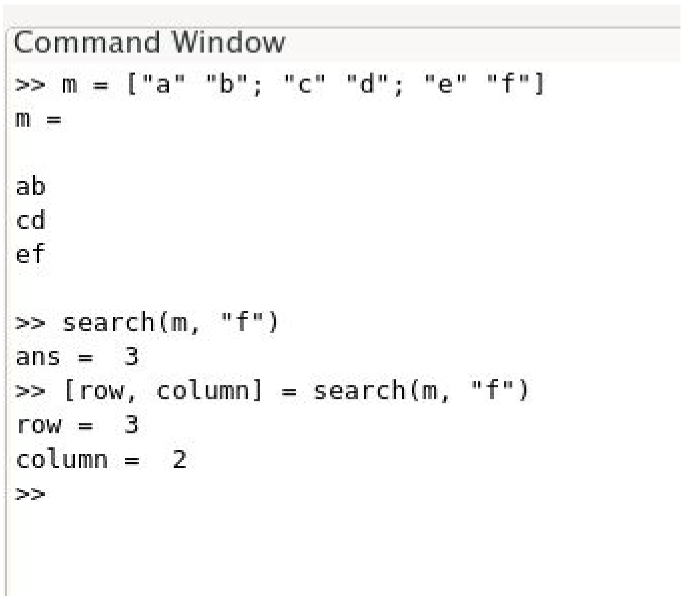


```
Command Window
>> m = [1 2 3; 4 5 6; 7 8 9]
m =

   1   2   3
   4   5   6
   7   8   9

>> eigen(m)
ans =  15.000
>> |
```

**5. Given a vector of unique names, and given one particular name that has to be found, write a function that searches for that name from the vector and returns**

**the element number. Proceed to returning row/column number from matrix of names.**

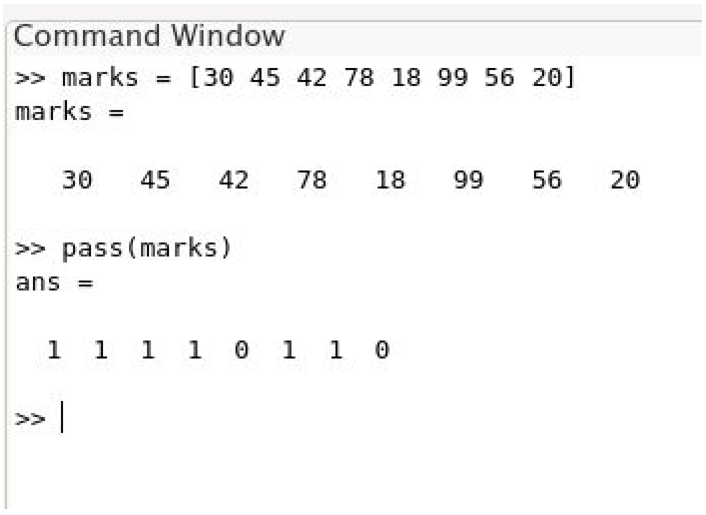
```
function [r c] = search(matrix, string)
  [R C] = size(matrix);
  r = 0;
  c = 0;
  for i = 1:R
    for j = 1:C
      if matrix(i, j) == string
        r = i;
        c = j;
        break
      end
    end
  end
end
```



```
>> m = ["a" "b"; "c" "d"; "e" "f"]
m =

ab
cd
ef

>> search(m, "f")
ans =  3
>> [row, column] = search(m, "f")
row =  3
column =  2
>>
```

**6. Write a function that takes a vector that contains the marks of students and returns the pass/fail status. Any score that is greater than, or equal to 30 is considered to be a passing score. Return a vector of the same size as the input vector, but has a particular element as 1, if the corresponding student passed, and 0 otherwise.**

```
function result = pass(vector)
  result = vector >= 30;
end
```



```
>> marks = [30 45 42 78 18 99 56 20]
marks =

   30   45   42   78   18   99   56   20

>> pass(marks)
ans =

  1  1  1  1  0  1  1  0

>> 
```