MCQ

1. 2
2. 2
3. 1
4. 1
5. 4
6. 2
7. 2
8. 2
9. 1
10. 2
11. 5
12. 2
13. 3
14. 5

Coding-on-paper Problems (MATLAB-like syntax)

15.
```
% Vectorized MATLAB version of the Python script

% ----------------------------
% Inputs / endpoints
% ----------------------------
x1 = 5000;  y1 = -2000;          % m
x2 = 7000;  y2 =  3000;          % m

r1 = [x1, y1];                   % 1x2
r2 = [x2, y2];                   % 1x2

% Distance between endpoints
d = sqrt(sum((r2 - r1).^2));     % m

% ----------------------------
% Time discretization
% ----------------------------
N  = 100;
v  = 5;                          % m/s
T  = d / v;                      % s
dt = T / N;                      % s

k    = 0:(N-1);                  % matches np.arange(N)
time = k * dt;                   % 1xN

% ----------------------------
% Trajectory (N x 2), vectorized
% ----------------------------
dir = (r2 - r1) / d;             % 1x2 unit direction vector

% array broadcasted across time and direction
r = r1 + (v * time(:)) .* dir;   % (N x 1).*(1 x 2) -> N x 2

x = r(:,1);                      % N x 1
y = r(:,2);                      % N x 1

% ----------------------------
% Concentration along path (vectorized plume)
% ----------------------------
conc = plume_vec(x, y);          % mg/m^3, N x 1

% ----------------------------
% Exposure in mg/m^3 * hr
% ----------------------------
exposure = sum(conc) * dt / 3600;


% ----------------------------
% Extra: formatted output
% ----------------------------
```

```
% Display result
fprintf('Exposure = %.6g (microg/m^3)*h\n', exposure);

% Plot concentration profile
plot(time ./ 60, conc ./ 1e3)
xlabel('time (minutes)')
ylabel('concentration (mg/m^3)')

% ========================================================================
% Vectorized plume function (accepts x,y vectors of same size)
% ========================================================================
function C_units = plume_vec(x, y)
    % x, y: vectors (or arrays) of the same size [m]
    % returns concentration in microg/m^3

    % Source / met parameters
    Q = 60;       % kg/s  (note: renamed from q to avoid collision)
    H = 60;       % m
    u = 5;        % m/s

    % Dispersion parameters (as in the Python code)
    a = 0.11;
    p = 0.08;
    q = 0.0002;  % renamed from q (Python) to avoid overwriting Q
    r = -0.5;

    % Sigma-y and sigma-z (vectorized)
    sy = a .* x .* (1 + 0.0001 .* x).^(-0.5);  % m
    sz = p .* x .* (q .* x).^r;                % m

    % Concentration (vectorized)
    C = Q ./ (2*pi*u .* sy .* sz) ...
        .* exp(-(y.^2) ./ (2*sy.^2)) ...
        .* exp(-(H.^2) ./ (2*sz.^2));

    C_units = C * 1e9;                         % microg/m^3
end
```

16.
```
% ----------------------------
% Read file
% ----------------------------
[year, month, day, hour, minute, direction, speed] = ...
    readfile('LAUSANNE_wind.csv');

% ----------------------------
% Iterate over months
% ----------------------------
letter = 'NESW';
for i = 1:12
    cond = month == i & speed > 2;

    % assign number of observations in each quadrant
    counts = zeros(1, 4);
    counts(1) = length(find(cond & (direction > 315 | direction <= 45)));
    counts(2) = length(find(cond & (direction > 45  & direction <= 135)));
    counts(3) = length(find(cond & (direction > 135 & direction <= 225)));
    counts(4) = length(find(cond & (direction > 225 & direction <= 315)));

    % find direction most frequent
    [maxval, maxpos] = max(counts);

    % print output
    fprintf('month = %d, direction = %s\n', i, letter(maxpos));
end

% ========================================================================
% File reader
% This is the actual function that would have been provided
% ========================================================================
function [year, month, day, hour, minute, direc, speed] = readfile(filename)
inp = readmatrix(filename);
year = int16(inp(:,1));
month = int16(inp(:,2));
```

```
day = int16(inp(:,3));
hour = int16(inp(:,4));
minute = int16(inp(:,5));
direc = inp(:,6);
speed = inp(:,7);
end
```