Pro1

```
x=pi/4
LHS=cos(x/2)^2;
RHS=(tan(x) + sin(x))/2*tan(x);
fprintf("left hand side:%.5f\n",LHS);
fprintf("right hand side:%.5f\n",RHS);
if abs(LHS-RHS)<1e-10
   disp('the identity is verified');
else
   disp('the identity is not verified');
end
```

Pro2

```
t0=120;
ts=38;
k=0.45;
t=3;
T=ts+(t0-ts)*exp(-k*t);
T=round(T);
disp(T);
```

Pro 3

```
m=[2, 4, 5, 10, 20, 50];
F=[12.5, 23.5, 30, 61, 117, 294];
g=9.81;
mu=F ./ (m*g);
average_mu = mean(mu);
disp('coefficient of friction for each test:');
disp(mu);
disp({'average coefficient of friction:',num2str(average_mu)});
```

pro 4

```
n = 500; r = 0.1; l = 0.4; omega = 2 * pi * n / 60;

theta = linspace(0, 2*pi, 1000);

x = r * (1 - cos(theta)) + (l - sqrt(l^2 - (r * sin(theta)).^2));

v = gradient(x, theta) * omega;

a = gradient(v, theta) * omega;


% Display results

fprintf('Theta: %.2f rad, Position: %.4f m, Velocity: %.4f m/s, Acceleration: %.4f m/s^2\n', ...

    [theta; x; v; a]);


% Plot

figure;

subplot(3, 1, 1); plot(theta, x, 'r'); title('Piston Position'); xlabel('\theta'); ylabel('Position (m)');

subplot(3, 1, 2); plot(theta, v, 'b'); title('Piston Velocity'); xlabel('\theta'); ylabel('Velocity (m/s)');

subplot(3, 1, 3); plot(theta, a, 'g'); title('Piston Acceleration'); xlabel('\theta'); ylabel('Acceleration
(m/s^2)');
```

pro5

```
v0 = 250; theta_deg = 65; wind_speed = 30; g = 9.81;

theta_rad = deg2rad(theta_deg);

t_flight = 2 * v0 * sin(theta_rad) / g;

t = linspace(0, t_flight, 1000);


% Calculate positions

x_no_wind = v0 * cos(theta_rad) * t;

y_no_wind = v0 * sin(theta_rad) * t - 0.5 * g * t.^2;

x_with_wind = x_no_wind + wind_speed * t;


% Plot
```

```
figure;
plot(x_no_wind, y_no_wind, 'b', x_with_wind, y_no_wind, 'r--', 'LineWidth', 1.5);
xlabel('Horizontal Distance (m)'); ylabel('Vertical Distance (m)');
title('Projectile Trajectories'); legend('Without Wind', 'With Wind');
grid on;
```

pro6

```
balance = 300000; interest_rate = 0.05; inflation_rate = 0.02; withdrawal = 25000;
years = 0; balance_history = []; withdrawals = [];

while balance > 0
    years = years + 1;
    balance_history(end+1) = balance;
    withdrawals(end+1) = withdrawal;
    balance = balance * (1 + interest_rate) - withdrawal;
    withdrawal = withdrawal * (1 + inflation_rate);
end

balance_history(end+1) = balance; withdrawals(end+1) = withdrawal;

% Plot
plot(0:years, balance_history, 'b', 0:years, withdrawals, 'r', 'LineWidth', 1.5);
xlabel('Years'); ylabel('Amount ($)'); title('Retirement Savings and Withdrawals');
legend('Balance', 'Withdrawals'); grid on;
```

Pro 7

```
%% Random Order of Singers
singers = {"John", "Mary", "Tracy", "Mike", "Katie", "David"};
random_order = singers(randperm(length(singers)));
disp(random_order);
```

pro 8

```
%% Function for Projectile Trajectory
function [max_height, max_distance] = projectile_trajectory(v0, angle)
    g = 9.81;
    angle_rad = deg2rad(angle);
    t_flight = 2 * v0 * sin(angle_rad) / g;
    t = linspace(0, t_flight, 100);
    x = v0 * cos(angle_rad) * t;
    y = v0 * sin(angle_rad) * t - 0.5 * g * t.^2;
    max_height = (v0^2 * sin(angle_rad)^2) / (2 * g);
    max_distance = v0^2 * sin(2 * angle_rad) / g;
    figure;
    plot(x, y, 'b', 'LineWidth', 1.5);
    xlabel('Distance (m)'); ylabel('Height (m)');
    title('Projectile Trajectory'); grid on;
end

% Example Usage
[max_height, max_distance] = projectile_trajectory(230, 39);
fprintf('Max Height: %.2f m, Max Distance: %.2f m\n', max_height, max_distance);
```

Pro9

```
% Given values
length_outside = 24; % inches
width_outside = 12; % inches
height_outside = 4; % inches
specific_weight = 0.101; % lb/in^3
weight_target = 15; % lb

% Function to calculate weight based on wall thickness x
calc_weight = @(x) specific_weight * ( ...
    (length_outside * width_outside) + ...
    2 * (length_outside + width_outside) * height_outside - ...
    (length_outside - 2*x) * (width_outside - 2*x) - ...
    2 * (length_outside + width_outside - 4*x) * (height_outside - x));

% Find thickness x using fminsearch
x_solution = fminsearch(@(x) abs(calc_weight(x) - weight_target), 0.1);

% Display the result
disp(['The thickness x is ', num2str(x_solution), ' inches']);
```