```matlab
%%%%%%%%%%%%%% Main Entrance %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% By Max Yi Ren and Emrah Bayrak %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Instruction: Please read through the code and fill in blanks
% (marked by ***). Note that you need to do so for every involved
% function, i.e., m files.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Optional overhead
clear; % Clear the workspace
% Note: for debugging purpose, do not use "clear all"
close all; % Close all windows
clc; %Clear screen

%% Optimization settings
% Here we specify the objective function by giving the function handle to a
% variable, for example:
f = @(x)distance(x); % replace rosenbrock with your objective function
% In the same way, we also provide the gradient and the Hessian of the
% objective:
g = @(x)distanceg(x); % replace accordingly
H = @(x)distanceH(x); % replace accordingly
% Note that explicit gradient and Hessian information is only optional.
% However, providing these information to the search algorithm will save
% computational cost from finite difference calculations for them.

% Specify algorithm
opt.alg = 'gradient';
%opt.alg = 'newton';

% Turn on or off line search. You could turn on line search once other
% parts of the program are debugged.
opt.linesearch = true; % or true

% Set the tolerance to be used as a termination criterion:
opt.eps = 1e-4; % this should be a small number like 1e-3

% Set the initial guess:
x0 = [2; 3]; % this should be a p-dim vector where p is the size of the
% problem

%% Run optimization
% Run your implementation of the gradient descent and Newton's method. See
% gradient.m and newton.m.
if strcmp(opt.alg,'gradient')
    solution = gradient(f,g,H,x0,opt);
elseif strcmp(opt.alg,'newton')
    solution = newton(f,g,H,x0,opt);
end

%% Report
% Implement report.m to generate a report.
report(solution,f);
```