1. Write a method with signature

```
function x_sol, res = grad_descent(f,gradf,x0)
```

The inputs `f` and `gradf` are function handles. The function `f`: $\mathbb{R}^N \to \mathbb{R}$ is an arbitrary objective function, and `gradf`: $\mathbb{R}^N \to \mathbb{R}^N$ is its gradient. The method should minimize f using gradient descent, and terminate when the gradient of f is small. I suggest stopping when

$$\|\nabla f(x^k)\| < 10^{-4}\|\nabla f(x^0)\|.$$

Use a backtracking line search to guarantee convergence. The stepsize should be monotonically decreasing. Each iteration should begin by trying the stepsize that was used on the previous iteration, and then backtrack until the Armijo condition holds:

$$f(x^{k+1}) \leq f(x^k) + \alpha\langle x^{k+1} - x^k, \nabla f(x^k)\rangle,$$

where $\alpha \in (0,1)$, and $\alpha = 0.1$ is suggested.

The function returns the solution vector `x_sol`, and a vector `res` containing the norm of the residual (i.e., the gradient in this case) at each iteration.

You can obtain the initial stepsize by estimating a Lipschitz constant for `f` using the formula:

$$L = \frac{\|\nabla f(x) - \nabla f(y)\|}{\|x - y\|}$$

where $x = x0$ and y is obtained by adding a small random perturbation to x. The initial stepsize is then $\tau = \frac{2}{L}$. Use a vector of zeros as the initial guess x0.

What happens if you don't use the Armijo linesearch? Does the method still converge?

2. Modify your solution from Question 1 to create the new function

```
function D, c = grad_descent_BB(f,gradf,x0)
```

This method is the same as the method from problem 1. However, instead of using a monotonically decreasing stepsize, begin each linesearch with a Barzilai-Borwein stepsize of magnitude

$$\tau = \frac{\langle x^{k+1} - x^k, x^{k+1} - x^k\rangle}{\langle x^{k+1} - x^k, \nabla f(x^{k+1}) - \nabla f(x^k)\rangle}.$$

3. Modify your solution from Question 1 to create the new function

```
function D, c = grad_descent_nesterov(f,gradf,x0)
```

This function should implement Nesterov's method:

$$\begin{aligned} x^k &= y^k - \tau\nabla f(y^k) \\ \delta^{k+1} &= \frac{1 + \sqrt{1 + 4(\delta^k)^2}}{2} \\ y^{k+1} &= x^k + \frac{\delta^k - 1}{\delta^{k+1}}(x^k - x^{k-1}). \end{aligned}$$

The method is initialized with $x^0 = y^1$ and $\alpha^1 = 1$, and the first iteration has index $k = 1$. Use the monotone line search from Question 1. For the Nesterov problem, what backtracking condition is

$$f(x^k) \leq f(y^k) + \alpha\langle x^k - y^k, \nabla f(y^k)\rangle.$$

Unlike standard gradient methods, Nestorov's method requires $\alpha \in [1/2, 1)$. I suggest using $\alpha = 1/2$.

4. Test your three solvers using the logistic regression classification problem from homework 1. Create a sample classification problem with 200 feature vectors and 20 features per vector using the method `create_classification_problem`. Then use your solvers to minimize `logreg_objective` using the gradient function `logreg_grad`. Plot the residuals for the different methods using a logarithmic y-axis. For example, you could call

```
[D, c] = create_classification_problem(200, 20, 1);
[x_sol, res] = grad_descent( @(x) logreg_objective(x,D,c),
                             @(x) logreg_grad(x,D,c), x0);
semilogy(res);
```

Create a short writeup (pdf) containing a plot comparing the 3 different algorithm. Which method was best? Does convergence speed depend on the problem dimension? Does it depend on how linearly separable the data are?