Question 1
Write a function that returns the linear approximation of a function passed to it. The function should take as arguments:

- A function f mapping some interval $[a, b]$ into $\mathbb{R}$.
- Two scalars a, b providing the limits of this interval.
- An integer n determining the number of grid points.
- A number x satisfying $a \leq x \leq b$.

The function should return the piecewise linear interpolation of f at x. Hint: look up the *findfirst* function and use it to define a function that locates the index of the first element of an array that is bigger than x.

Question 2
In this problem, we consider how to optimally approximate a simple function using linear interpolation. Say that we have the function $f(x) = \ln(x + 1)$, with $x \in [0, 100]$. Our goal is to form the best possible approximation of this function with linear interpolation while using only a fixed number of grid points. Specifically, write a function that:

- Inputs a discretization of the [0, 100] range. The first element should be zero, the final element 100, and there should be $n = 9$ grid points in between. Generate an approximation $\hat{f}(x)$ by evaluating $f(x)$ at the grid points and then defining a linear interpolation of the discretized function.
- Evaluates the fit of this approximation by evaluating $|\hat{f}(x) - f(x)|$ from $x = [0, 100]$ in increments of 0.1. Returns vectors containing $f(x)$, $f(\hat{x})$, and the approximation error $|\hat{f}(x) - f(x)|$ for each 0.1 increment from 0 to 100.

A) Start with an evenly spaced grid (i.e. 0, 10, 20 . . .). Plot $f(x)$, $\hat{f}(x)$, and the approximation error. Where is the approximation error the highest? Interpret.

B) Now write a function that admits a vector of 9 points between 0 and 100. The function should then add on 0 and 100 to either end of the vector to form a new discretization of the domain of x before running the function above for the new grid. The output of the function should be the total approximation error (i.e. the sum of all $|\hat{f}(x) - f(x)|$).

C) Using Nelder-Mead with a starting guess of evenly-spaced points, find the optimal point placement for minimizing the total approximation error of the interpolated function. What are the selected grid points? By how much does the total approximation error decrease? Plot the new grid of approximation errors, $f(x)$, and $\hat{f}(x)$.

D) Repeat C) using a cubic splines. Describe how the point placement and total approximation error changes between the different specifications.

Question 3
The **Himmelblau function** is of the form

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2.$$

The critical points of this function can be found analytically, but doing so is not recommended. Instead, we will find the minima using optimizers.

A) Produce a surface plot of the function over the domain $x, y \in [-4, 4]$. How many local minima do there appear to be? Note: playing with the "camera()" option with Julia plots allows you to alter the angle from which you view the surface plot.

B) Solve for the gradient and Hessian of the function, and optimize it using Newton's method. Comment on how the answer you arrive at depends on the initial guess fed to the algorithm.

C) Now ignore the derivatives and optimize the function using Nelder-Mead. Compare the number of iterations required to arrive at local minima compared to Newton's method.

Question 4)
The **Ackley function** is a non-convex function used as a performance test problem for optimization algorithms. The function is defined by

$$f(x,y) = -20 \exp\left[-0.2\sqrt{0.5 \cdot (x^2 + y^2)}\right] - \exp[0.5 \cdot (\cos 2\pi x + \cos 2\pi y)] + e + 20$$

A) Produce surface and contour plots of the Ackley function over the domain $x, y \in [-4, 4]$. What is the global minimum?

B) Minimize the function using LBFGS and Nelder-Mead. Experiment with how the performance of the algorithms responds to the starting guess fed to them.

Question 5)
The **Rastrigin function** is a particularly absurd function used for testing optimization algorithms. On an n-dimensional domain, the problem is defined by

$$f(\boldsymbol{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right],$$

where $A = 10$ and $x_i \in [-5.12, 5.12]$.

A) Plot the function for $n = 1$. What is the global minimum? Does this minimum hold for any arbitrary n?

B) Produce surface and contour plots for the Rastrigin function for $n = 2$.

C) Minimize the function using LBFGS and Nelder-Mead. Experiment with how the performance of the algorithms responds to the starting guess fed to them.

Question 6)
This question will take you through estimating a normal linear model using maximum likelihood. You will also conduct inference using three methods. The *lwage.csv* dataset contains a large sample of men aged 25-65 in the 2000 Census. The variables include log hourly wage, a dummy for college attainment, and years of experience in the labor force. Suppose the data-generating process is $y_i = \beta_0 + \beta_1 c_i + \beta_2 x_i + \beta_3 x_i^2 + \varepsilon_i$, where c_i, x_i denote college attainment and experience for individual i, y_i is log hourly wage, and ε_i is an error term distributed $N(0, \sigma^2)$ and is assumed exogenous.

A) Estimate the model using OLS in Stata. Report standard errors and t-statistics.

B) Derive the log-likelihood of the model for any arbitrary guess of parameters $(b_0, b_1, b_2, b_3, s^2)$, where s^2 is a guess of the variance of the error term. Hint: recall that the PDF for $N(0, \sigma^2)$ is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}$$

C) Estimate the model via log likelihood and report your results. I recommend following the example set at https://julianlsolvers.github.io/Optim.jl/stable/#examples/generated/maxlikenlm/.

D) We will now obtain standard errors using a Bootstrap method. For a single bootstrap run, you will keep only half the dataset and then obtain parameter estimates according to the procedure in part C). Do this 100 times to obtain 100 different estimates of the model parameters before reporting the sample standard errors of the parameter estimates. Each time you do this, use a different random seed so as to induce randomness in which observations are kept and which are left out. Run this program on a single core, and then parallelize it to run on Linstat with 20 cores. Comment on how much time parallelization saves here.