1. **What purpose does back-propagation serve?**

   Assume the weights as salt and biases as pepper that are mixed with the input, which is the soup. Our task is to construct a neural network, consider it as a black box that determines how much salt/pepper should be added so that the soup will taste better. We begin by constructing a neural network with arbitrary salt and pepper. We then taste the soup generated by our network and adjust the quantity of salt/pepper to get the required taste. This process is called backpropagation. It means that we compare the generated output of a neural network with the intended output and feed the error to the network to find the weights and biases for which minimize the error is minimum.

2. **Can you describe the back-propagation in a feedforward neural network that's trying to learn?**

   The cost/loss/error, which is the difference between the actual output and predicted output, is a function of weights and biases. We need to find the right value for the weights and biases for which the cost is minimal. Optimization algorithms like gradient descent help to perform this. Backpropagation is a technique that helps to determine how a single training example wishes to update the weights and biases. The process has to be repeated for all training examples to arrive at the optimal weight and bias.

   The activation or the value held by a neuron, say A, can be mathematically expressed as, A = Activation Function (W*(A-1) +b), where W is the weight, A-1 is the activation of the previous layer which acts the input to the current layer and b is the bias. For instance, if a neuron has an activation of 0.4 instead of 0.9, we would like to increase it. This can be done by increasing the weights (W) relative to the previous layer activations (A-1), increasing the biases or increasing the previous layer activations (A-1) relative to the current layer weights (W). The activation of the previous layer can't be increased directly as it is determined by a different set of inputs which are specific to the previous layer. Mathematically, A-1 = Activation Function (W_New*(A-2)+b_New). In short, the value/activation of the neurons in the output layer is determined by the neurons in the penultimate layer. This dependency propagates backwards and in order for an output neuron to have a certain activation/value, the activations of almost all neurons in each layer of the network should be changed. This in turn means that the weights and biases associated with each connection between neurons has to be changed. Since the change/updation of weights/biases propagates backwards, we call it backpropagation.

3. **Can you explain the back-propagation process for deep-learning in terms of the analogy?**

   Consider various stages (layers in a neural network) that perform certain tasks to covert the raw ingredients (inputs) to soup (output). In each stage, the ingredients are cooked with certain arbitrary amount of salt(weights) and pepper(pepper). We consider the back-propagation as the process of correcting the salt (weights) and pepper (biases) based on the soup's taste(error). Better the soup's taste the lower the error. The salt and pepper that contribute to the taste of the soup in the last stage is determined by the salt/pepper that is added not only in the last stage but also that in earlier stages. So after the soup's taste(error) is not as what we expect we try to bring it close to the required value by adjusting the salt/pepper in earlier stages with reference to the present stage.

   Assume that the soup at the end of last stage is salty but we require a soup that has the right amount of all spices. We can do this by adding more pepper in the last stage or adding no salt in the last stage or by correcting the salt of the soup that is added to the last stage by the previous stage. The last method of correcting the salt in the previous stage can't be done

directly. This requires going back to that stage and looking at ways to reduce the salt. This process is done for all the stages until the soup's taste (error) is corrected.

4. **With or without an analogy, but in your own words, describe the gradient descent algorithm.**

   Gradient descent algorithm is an optimization algorithm to minimize the cost/loss/error function, which depends on weights and biases. First, we need to estimate the error in the prediction (cost function). Then the gradient of the current cost with respect to the weights and biases has to be found. The gradient will give us an idea of how sensitive the cost is for changes in a particular weight/bias. This information is used to determine whether the weight/bias needs to be increased/decreased and by what amount. One parameter that determines the extent of change in the weights and biases is the learning rate of the optimization algorithm. Once the weights/biases are updated, we need to predict again and calculate the new error. Repeat this process until we reach the local minima of the function i.e. the cost does not decrease with further change in weights/biases.

5. **Imagine and state a problem that you could use gradient descent on**

   Gradient descent can be used for predicting sales of an item given the past sales history.

6. **Can gradient descent help for problems related to sequences over finite fields?**

   I think yes. I am not sure of the justification of my answer.