1. Let $\mathcal{M}_1$ be an ε_1-differentially private mechanism, and let $\mathcal{M}_2$ be an ε_2-differentially private mechanism. Argue that the sequential composition theorem, which states that the combination of $\mathcal{M}_1$ and $\mathcal{M}_2$, given by $\mathcal{M}_{1,2} = (\mathcal{M}_1, \mathcal{M}_2)$, is $\varepsilon_1 + \varepsilon_2$-differentially private. Describe the intuition behind this privacy guarantee.

   Solution. Recall the definition of an ε-differentially private mechanism:

   $$\Pr[\mathcal{M}(x) \in S] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in S].$$

   Let r_1 be some output of $\mathcal{M}_1$ and r_2 be some output of $\mathcal{M}_2$. By definition,

   $$\Pr[\mathcal{M}_1(x) = r_1] \leq \exp(\varepsilon_1) \Pr[\mathcal{M}_1(y) = r_1]$$

   and

   $$\Pr[\mathcal{M}_2(x) = r_2] \leq \exp(\varepsilon_2) \Pr[\mathcal{M}_2(y) = r_2].$$

   Then we have

   $$\begin{aligned}\frac{\Pr[\mathcal{M}_{1,2}(x) = (r_1, r_2)]}{\Pr[\mathcal{M}_{1,2}(y) = (r_1, r_2)]} &= \frac{\Pr[\mathcal{M}_1(x) = r_1] \Pr[\mathcal{M}_2(x) = r_2]}{\Pr[\mathcal{M}_1(y) = r_1] \Pr[\mathcal{M}_2(y) = r_2]} \\ &= \left(\frac{\Pr[\mathcal{M}_1(x) = r_1]}{\Pr[\mathcal{M}_1(y) = r_1]}\right)\left(\frac{\Pr[\mathcal{M}_2(x) = r_2]}{\Pr[\mathcal{M}_2(y) = r_2]}\right) \\ &= \exp(\varepsilon_1)\exp(\varepsilon_2) \\ &= \exp(\varepsilon_1 + \varepsilon_2).\end{aligned}$$

   This gives us

   $$\Pr[\mathcal{M}_{1,2}(x) = (r_1, r_2)] = \exp(\varepsilon_1 + \varepsilon_2) \Pr[\mathcal{M}_{1,2}(y) = (r_1, r_2)].$$

   Thus, $\mathcal{M}_{1,2}$ is $\varepsilon_1 + \varepsilon_2$-differentially private.

   The intuition behind this conclusion is that each time a data set is queried via a particular mechanism, more is learned about that data and thus the privacy loss increases. The sequential composition theorem allows us to bound the privacy loss when more than one differentially private mechanism is applied on the data. This is an upper bound, i.e., the actual total privacy loss could be less.

2. Let $\mathcal{M}$ be an ε-differentially private mechanism, and let X be a data set that has been split into k disjoint chunks such that $X = x_1 \cup x_2 \cup \cdots \cup x_k$. Argue that the parallel composition theorem, which states that the combination of $\mathcal{M}(x_1), \mathcal{M}(x_2), \cdots, \mathcal{M}(x_k)$ is ε-differentially private. Describe the intuition behind this privacy guarantee.

   Solution. Let r_i be some output of $\mathcal{M}(x_i)$. By the definition of differential privacy we have

   $$\begin{gathered}\Pr[\mathcal{M}(X) = r] = \exp(\varepsilon) \Pr[\mathcal{M}(Y) = r] \\ \Pr[(\mathcal{M}(x_1), \cdots, \mathcal{M}(x_k)) = (r_1, \cdots, r_k)] = \exp(\varepsilon) \Pr[(\mathcal{M}(y_1), \cdots, \mathcal{M}(y_k)) = (r_1, \cdots, r_k)]\end{gathered}$$

   Using the definition of joint probability we get

   $$\prod_i \Pr[\mathcal{M}(x_i) = r_i] = \exp(\varepsilon) \prod_i \Pr[\mathcal{M}(y_i) = r_i].$$

Thus $\prod_i \Pr[\mathcal{M}(x_i) = r_i]$ is ε-differentially private. This bound for the privacy cost is better than that given by the sequential privacy theorem, which would be $k\varepsilon$.

The intuition behind this conclusion is that $\mathcal{M}$ queries each chunk in X one time each, so the total privacy cost will not exceed ε.

3. The Laplace distribution centered at μ with scale b is given by the following probability density function for some random variable X:

$$\text{Lap}(b, \mu)(X) = \frac{1}{2b} \exp\left(-\frac{|\mu - X|}{b}\right).$$

Recall that the Laplace mechanism samples from the Laplace distribution with scale $\frac{\Delta}{\varepsilon}$, where Δ is the sensitivity of the function f to which we are adding Laplace noise. Argue that the Laplace mechanism preserves ε-differential privacy.

Solution. Let $\mathcal{M}_L(D, f, \varepsilon)$ represent the Laplace mechanism for a function $f(D)$ with privacy budget ε, where D is some input dataset. Let D' be another dataset which differs from D by a single element. We compare their probability density functions for some arbitrary output $z \in \mathbb{R}$.

$$\begin{aligned}
\frac{\Pr[\mathcal{M}_L(D, f, \varepsilon) = z]}{\Pr[\mathcal{M}_L(D', f, \varepsilon) = z]} &= \frac{\frac{1}{2b}\exp\left(-\frac{|f(D)-z|}{\Delta/\varepsilon}\right)}{\frac{1}{2b}\exp\left(-\frac{|f(D')-z|}{\Delta/\varepsilon}\right)} \\
&= \frac{\exp\left(-\frac{\varepsilon|f(D)-z|}{\Delta}\right)}{\exp\left(-\frac{\varepsilon|f(D')-z|}{\Delta}\right)} \\
&= \exp\left(\frac{\varepsilon(|f(D') - z| - |f(D) - z|)}{\Delta}\right) \\
&\leq \exp\left(\frac{\varepsilon|f(D') - f(D)|}{\Delta}\right) \\
&= \exp\left(\frac{\varepsilon\Delta}{\Delta}\right) \\
&\leq \exp(\varepsilon)
\end{aligned}$$

The first inequality follows from the triangle inequality, and the last line follows from the definition of the sensitivity of f.

4. Suppose we want to compute the average starting salary of a group of Rensselaer Polytechnic Institute computer science graduates. The salaries are given in the data set `D1`. Now suppose we remove one salary from the data set, given by the data sets `D2` through `D6`.

```
import numpy as np

D1 = np.array([130530, 104141, 90385, 91919, 81334])
D2 = np.array([104141, 90385, 91919, 81334])
D3 = np.array([130530, 90385, 91919, 81334])
D4 = np.array([130530, 104141, 91919, 81334])
D5 = np.array([130530, 104141, 90385, 81334])
D6 = np.array([130530, 104141, 90385, 91919])
```

(a) Write a query `f(D)` that computes the average salary of a given data set.

Solution. The query should look like the following:

```
def f(D):
    return sum(D) / len(D)
```

(b) Compute the local sensitivity of `f` given `D1`.

Solution. The salary average that differs the most from `D1` is that of `D2`. Thus, the local sensitivity can be calculated as follows:

```
sensitivity = abs(f(D1) - f(D2))
```

(c) Let $\varepsilon = 1.0$ be the privacy loss of `f`. Write a new query K that is ε-differentially private by adding noise to `f`. Note that K is given by the Laplace mechanism for DP functions:

$$K(D) = f(D) + \text{Lap}\left(\frac{s}{\varepsilon}\right),$$

where s is the sensitivity of f.

Solution. The implementation for the updated query should look like the following:

```
def K(D, sensitivity, epsilon):
    return f(D) + np.random.laplace(loc = 0.0, scale = sensitivity
   / epsilon)
```

(d) Compute the average of `D1`, both with and without noise.

Solution. The averages with and without noise, respectively, can be calculated as follows:

```
f_D1 = f(D1)
K_D1 = K(D1, sensitivity, epsilon)
```

(e) Perform tuning of the privacy budget and compare the results. What is the relationship between the privacy loss, amount of noise added, and accuracy of the query? Discuss the differential privacy versus accuracy tradeoff.

Solution. Below is a table comparing different ε-differentially private query outputs:

ε	$f(D1)$
0.1	92788.53
1.0	111458.11
2.0	101603.66
10.0	99400.37
100.0	99682.58
1000.0	99675.37

The higher the privacy budget ε, the more accurate the outcome of a query. This is because the amount of noise that is added to an output is inversely proportional to ε. Thus, if ε is large, less noise will be added. Conversely, a lower acceptable privacy loss will result in more noise being added to better hide the data, and as a result the output will be less accurate.

5. Suppose we perform 1,000 independent runs of a (ε, δ)-differentially private algorithm with $\varepsilon = 0.4$ and $\delta = 10^{-6}$. Compute the privacy budget of this algorithm using (a) the sequential composition theorem and (b) the strong composition theorem. Now suppose that each run samples a batch of 1 percent of the data. Compute new bounds for the privacy loss and failure probability using (c) the strong composition theorem with amplification and (d) the moments accountant. Compare the guarantees of each theorem.

   Solution. The guarantees for privacy loss and failure probability are as follows:

   (a) $\varepsilon_{\text{total}} = T\varepsilon = 1000 \times 0.4 = \boxed{400.0}$
   $\delta_{\text{total}} = T\delta = 1000 \times 10^{-6} = \boxed{0.001}$

   (b) $\varepsilon_{\text{total}} = \varepsilon\sqrt{T \log(\frac{1}{\delta})} = 0.4 \times \sqrt{1000 \times \log(\frac{1}{10^{-6}})} = \boxed{30.98}$
   $\delta_{\text{total}} = T\delta = 1000 \times 10^{-6} = \boxed{0.001}$

   (c) $\varepsilon_{\text{total}} = q\varepsilon\sqrt{T \log(\frac{1}{\delta})} = 0.01 \times 0.4 \times \sqrt{1000 \times \log(\frac{1}{10^{-6}})} = \boxed{0.3098}$
   $\delta_{\text{total}} = qT\delta = 0.01 \times 1000 \times 10^{-6} = \boxed{0.00001}$

   (d) $\varepsilon_{\text{total}} = q\varepsilon\sqrt{T} = 0.01 \times 0.4 \times \sqrt{1000} = \boxed{0.1265}$
   $\delta_{\text{total}} = \delta = \boxed{10^{-6}}$

   The sequential composition theorem has a very loose privacy loss guarantee, and this overestimation of ε will require us to add more noise than necessary to counter the privacy loss. The strong composition theorem is tighter, but we can do better if sample a subset of the data (instead of accessing the entire dataset) on each run. By applying the privacy amplification theorem, we get a privacy bound that is 100 times smaller. Still, we can do better with the moments accountant, which has the tightest privacy loss guarantee.

6. Tune hyperparameters for DP-SGD on the standard handwritten digits MNIST data set, to achieve $(2.0, 10^{-5})$-differential privacy using 50 epochs. Use a single layer logistic regression model with forward and backwards pass given by the following:

```
# Forward pass
scores = np.dot(X_batch, w.T) + b
probs = 1 / (1 + np.exp(-scores))
loss = -1/batch_size * \
    np.sum(y_batch * np.log(probs) + (1-y_batch) * np.log(1-probs))

# Backward pass
dscores = 1/batch_size * (probs - y_batch)
grad_w = np.dot(dscores.T, X_batch)
grad_b = np.sum(dscores, axis=0)
```

   Choose learning rate, C, σ, and batch size. What is the final accuracy and epsilon values? Use the Tensorflow Privacy function `compute_do_sgd_privacy`.

```
from privacy.analysis import compute_dp_sgd_privacy
```

   Plot test accuracy versus different values of C. How does choice of gradient-clipping parameter C affect the model?

Solution. An accuracy of 0.76 and privacy loss of 2.04 were achieved with the following tuned parameters: `learning_rate` $= 0.01$, `num_epochs` $= 50$, `batch_size` $= 100$, and $\sigma = 0.9$.

The higher the gradient clipping parameter C, the better the accuracy you get, as the cutoff for the gradient is higher (the gradient is less clipped and closer to what it would be without DP).

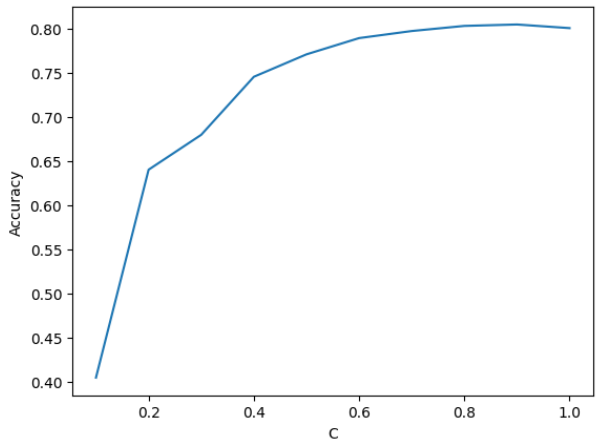

Figure 1: Test accuracy vs. gradient clipping C ($\delta = 10^{-5}$, $\sigma = 0.9$)