Lab 7 Solution key

1.

```
import matplotlib.pyplot as plt
import numpy as np

# Data points
X = np.array(
    [
        [1, 2],
        [1.5, 1.8],
        [5, 8],
        [8, 8],
        [1, 0.6],
        [9, 11],
        [8, 2],
        [10, 2],
    ]
)

# Seed centroids
np.random.seed(0)
centroids = X[np.random.choice(len(X), 2, replace=False)]

for _ in range(100):
    # Compute distances & assign clusters
    dists = np.linalg.norm(X[:, np.newaxis] - centroids, axis=2)
    labels = np.argmin(dists, axis=1)

    # Compute new centroids
    new_centroids = np.array([X[labels == k].mean(axis=0) for k in range(2)])
    if np.all(centroids == new_centroids):
        break
    centroids = new_centroids

print("Final Centroids:\n", centroids)

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap="coolwarm")
plt.scatter(
    centroids[:, 0], centroids[:, 1], color="black", marker="x", s=200
)
plt.show()
```

2.

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y_actual = iris.target

kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap="viridis")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("K-Means Iris Clusters")
plt.show()
```

3.

```
from collections import Counter
import numpy as np

# Training data
X = np.array([[1.0, 10], [2.0, 7], [3.0, 5], [4.0, 2]])
y = np.array(["Fail", "Fail", "Pass", "Pass"])

new_student = np.array([2.5, 5])

# Euclidean Distances
dists = np.linalg.norm(X - new_student, axis=1)

# Select K=3 nearest
k_indices = np.argsort(dists)[:3]
k_labels = y[k_indices]

prediction = Counter(k_labels).most_common(1)[0][0]

for idx in k_indices:
    print(f"Neighbor {idx}: Distance = {dists[idx]:.4f}, Label = {y[idx]}")
print("Predicted Label:", prediction)
```

4.

```
import numpy as np

X = np.array([1000, 1500, 2000, 2500])
```

```
y = np.array([150, 200, 250, 300])

target_size = 1800

# Euclidean Distance
dists = np.abs(X - target_size)

# K=2 nearest
k_indices = np.argsort(dists)[:2]
pred_price = np.mean(y[k_indices])

print(f"2-Nearest Prices: {y[k_indices]}")
print(f"Predicted Price: ${pred_price:.2f}k")
```

5.

```
from sklearn.datasets import fetch_california_housing
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor

housing = fetch_california_housing()
X_tr, X_te, y_tr, y_te = train_test_split(
    housing.data, housing.target, test_size=0.2, random_state=42
)

knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_tr, y_tr)

preds = knn.predict(X_te)
print("MSE:", mean_squared_error(y_te, preds))
print("R2 Score:", r2_score(y_te, preds))
```