DSC 190 - Discussion 03

Problem 1.

When performing a search for the k nearest neighbors to a query point, we need to keep track of the k smallest distances found so far. We can do so using a heap.

Fill in the class below so that it keeps track of the k smallest numbers inserted while maintaining a heap whose size is never larger than $k + 1$.

```
class KSmallest:

    def __init__(self, k):
        ...

    def insert(self, number):
        """Insert a number."""
        ...

    def max(self):
        """Return the largest of the k numbers stored."""

    def as_list(self):
        """Return the k elements as a list."""
        ...
```

Problem 2.

kNN search requires that we find the k nearest neighbours when we reach a leaf node in our search.

Fill in the brute force search function below to find k nearest neighbours to a point for a given leaf node.

```
def brute_force_knn_search(data, p, k):
    """
    Find nearest neighbour
    Parameters:
    data : np.ndarray
        An n X d array of points
    p : np.ndarray
        A d-array representing the query point
    k : int
        The number of neighbours to find

    Returns:
    knn : np.ndarray
        The k X d array of form [distance, point]
        where point is a d-array and distance is a float value
        represent distance to query point p
    """
```