1. Implement K-Means algorithm in MapReduce paradigm
   - Data – points (x,y) in [0,1]x[0,1]:
     0.72 0.44
     0.16 0.82
     0.42 0.37
     0.19 0.65
     …
   - Desired output:
     (0.72 0.44) (0.55 0.83)
     (0.16 0.82) (0.55 0.83)
     (0.42 0.37) (0.29 0.16)
     …

הסבר:
לחשב לכל נקודה מי הקלאסטר הקרוב אליה.
לאחר שיש לכל הנקודות, ב-reduce
לחשב ולפלוט את הקלאסטרים החדשים.
האלגוריתם יתבצע כאשר כל הקלאסטרים
לא מתעדכנים יותר.

Map:
```
def map(point, clusters):
    max = ∞
    cluster = -1
    for c in clusters:
        if distance(point, c) < max:
            cluster = c
            max = distance(point, c)
    yield (point, cluster)
```

Reduce:
```
def reduce(points, cluster)
    yield (mean(points), cluster)
```

2. Implement CheckClique function, that given an undirected graph returns if the graph is a full clique
   - Data:
     A-> B C
     B-> A C
     C-> A B
     ...
   - Desired output:
     YES

הרעיון:
לקחת כל צומת v, ולבדוק כמה
מסלולים שונים יש לו (לא כולל קשת
עצמית).
אם כל v מקיים |V|-1 אז
אז הגרף הוא קליקה

Map:
```
def map (v, adjacents):
    for n in adjacents and n ≠ v:
        yield(v,n)
```

Reduce:
```
def reduce (v, adjacents[])
    unique_neighbors = set (adjacents)
    yield (v, len(unique_neighbors))
```

3. Implement pseudo-synonyms detection algorithm in MapReduce paradigm

- Data – queries:

  buy cheap house

  buy big house

  buy new house

  rent cheap car

  rent new car

  rent Volkswagen car

  ...

- Desired output:

  cheap – new (2)

הרעיון:
לחשב איזה פונקציות של מ"מ.
הראשונה יפעל בזוגות של מילים
למילה שלישית, והשניה יפעל
על זוגות כבר לימנו אותן.
אם זוגות מסוימים יותר מספר אחת,
זוג כזה יחשב כ- pseudo-synonyms.

Map1:
```
def map1(sentences):
    words = sentences.split()
    yield ((words[0], words[2]), words[1])
```

Map2:
```
def map2(duo, words):
    for i in range(len(words)):
        for j in range(i+1, len(words)):
            yield ((words[i], words[j]), 1)
```

Reduce1:
```
def reduce1(duo, words):
    for word in words:
        yield (duo, word)
```

Reduce2:
```
def reduce2(duo, counts):
    score = sum(counts)
    if score > 1:
        yield (duo, score)
```