```
process( S₁: string, S₂: string):
  ---------------------------------
  N₁ = get_nouns( S₁ )  // List[str]
  N₂ = get_nouns( S₂ )  // List[str]
  ---------------------------------
  E₁ = get_edges( N₁ )  // List[tuple], len(E₁) = len(N₁) · (len(N₁) − 1)
  E₂ = get_edges( N₂ )  // List[tuple], len(E₂) = len(N₂) · (len(N₂) − 1)
  ---------------------------------
  best = 0
  for e₁ in E₁:
    for e₂ in E₂:
      best = max( best, get_score( e₁, e₂ ) )
  ---------------------------------
  return best
```

```
get_score( e₁: tuple, e₂: tuple):
  ---------------------------------
  props₁ = get_props( e₁ )  // List[str]
  props₂ = get_props( e₂ )  // List[str]
  ---------------------------------
  matches = []
  for p₁ in props₁:
    for p₂ in props₂:
      matches.append( similarity( p₁, p₂ ) )
  ---------------------------------
  matches = sorted(matches, reversed=True)[:5]
  return average(matches)
```

```
get_props( subject: string, object: string ):
  ---------------------------------
  props₁ = get_props_from_quasimodo( subject, object, n_largest = 10 )  // sorted by plausibility
  props₂ = get_props_from_google( subject, object )  // why do, why does, how do, how does
  ---------------------------------
  return props₁ + props₂
```

Examples:

```
1.  “On earth, the atmosphere protects us from the sun, but not enough so we use sunscreen”
2.  “The nucleus, which is positively charged, and the electrons which are negatively charged, compose the atom”
3.  --------------------------------
4.  Best match found:
5.  (earth → sun), (electrons → nucleus) --→ average score: 1.0
6.    - circle the ~ circle the → score: 1.0
7.    - revolve around ~ revolve around → score: 1.0
8.    - not fall into ~ not fall into → score: 1.0
9.    - orbit ~ orbit → score: 1.0
10.   - move around ~ move around → score: 1.0
```

```
1.  “A road is where cars are”
2.  “Boats sail on the lake to get from place to place
3.  --------------------------------
4.  Best match found:
5.  (cars → road), (boats → lake) --→ average score: 0.355
6.    - go on ~ be on → score: 0.648
7.    - go on ~ affect → score: 0.343
8.    - go on ~ travel through the great → score: 0.29
9.    - skid on wet ~ affect → score: 0.252
10.   - skid on wet ~ sink in a → score: 0.243
```

```
1.  “A singer expresses what he thinks by songs”
2.  “A programmer expresses what he thinks by writing code”
3.  --------------------------------
4.  Best match found:
5.  (singer → songs), (programmer → code) --→ average score: 0.819
6.    - remember ~ remember → score: 1.0
7.    - write ~ write → score: 1.0
8.    - write ~ write in → score: 0.728
9.    - can remember ~ remember → score: 0.693
10.   - remember ~ remember all → score: 0.674
```