```
import sys

damping = 0.85

page_links = {}
page_ranks = {}

for line in sys.stdin:
    page, value = line.strip().split("\t", 1)

    # If it's structure (list of links)
    if " " in value:
        page_links[page] = value
    else:
        # rank contribution
        val = float(value)
        page_ranks[page] = page_ranks.get(page, 0) + val

# Apply damping factor
for page in set(list(page_links.keys()) + list(page_ranks.keys())):
    rank_sum = page_ranks.get(page, 0.0)
    new_rank = (1 - damping) + damping * rank_sum
    links = page_links.get(page, "")
    print(f"{page}\t{new_rank}\t{links}")
```

```
import sys

for line in sys.stdin:
    parts = line.strip().split()
    if len(parts) == 0:
        continue

    page = parts[0]         # current page
    links = parts[1:]       # outgoing links
    num_links = len(links)

    # Emit structure info (to preserve graph structure)
    print(f"{page}\t{' '.join(links)}")

    # If page has rank (initial assume 1.0), distribute equally
    rank = 1.0
    if num_links > 0:
        contribution = rank / num_links
        for link in links:
            print(f"{link}\t{contribution}")
```

```
import sys

max_trailing = {}

for line in sys.stdin:
    word, tz = line.strip().split("\t")
    tz = int(tz)
    if word not in max_trailing:
        max_trailing[word] = tz
    else:
        max_trailing[word] = max(max_trailing[word], tz)

# FM estimate: 2^R
for word, r in max_trailing.items():
    print(f"{word}\tApproxFreq={2**r}")
```

```python
import sys
import hashlib

def custom_hash(word):
    # convert word → int
    x = int(hashlib.md5(word.encode()).hexdigest(), 16)
    return (2 * x + 3) % 10    # (2x+3)%10

def trailing_zeros(x):
    if x == 0:
        return 32    # arbitrary large (since FM assumes infinite zeros)
    tz = 0
    while (x & 1) == 0:
        tz += 1
        x >>= 1
    return tz

for line in sys.stdin:
    words = line.strip().split()
    for w in words:
        h = custom_hash(w)
        tz = trailing_zeros(h)
        print(f"{w}\t{tz}")
```

```python
import sys

# Bloom filter bit array of size 10
bit_array = [0] * 10

for line in sys.stdin:
    idx, _ = line.strip().split("\t")
    idx = int(idx)
    bit_array[idx] = 1   # set the bit at that position

# Print final Bloom filter bit array
print("BloomFilter:", " ".join(map(str, bit_array)))
```

```python
import sys

# custom hash using ascii sum
def custom_hash(word):
    x = sum(ord(c) for c in word)    # sum of ASCII values
    return (2 * x + 3) % 10          # (2x+3) % 10 → bucket index

for line in sys.stdin:
    words = line.strip().split()
    for w in words:
        h = custom_hash(w)
        # Emit index where this word maps
        print(f"{h}\t1")
```

```python
import sys

coolest_year = None
min_temp = float("inf")

for line in sys.stdin:
    year, temp = line.strip().split("\t")
    temp = int(temp)

    if temp < min_temp:
        min_temp = temp
        coolest_year = year

print(f"Coolest Year: {coolest_year}, Temperature: {min_temp}")
```

```
import sys

for line in sys.stdin:
    parts = line.strip().split()
    if len(parts) != 2:
        continue
    year, temp = parts
    print(f"{year}\t{temp}")
```

```
import sys

current_bigram = None
current_count = 0

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    parts = line.split('\t')
    if len(parts) != 2:
        continue
    bigram, count = parts
    try:
        count = int(count)
    except ValueError:
        continue
    if bigram == current_bigram:
        current_count += count
    else:
        if current_bigram:
            print("%s\t%d" % (current_bigram, current_count))
        current_bigram = bigram
        current_count = count

if current_bigram:
    print("%s\t%d" % (current_bigram, current_count))
```

```
import sys

for line in sys.stdin:
    words = line.strip().split()
    for i in range(len(words) - 1):
        bigram = "%s %s" % (words[i], words[i+1])
        print("%s\t1" % bigram)
```

```python
import sys

current_word = None
current_count = 0

for line in sys.stdin:
    word, count = line.strip().split('\t')
    count = int(count)
    if word == current_word:
        current_count += count
    else:
        if current_word:
            print("%s\t%d" % (current_word, current_count))
        current_word = word
        current_count = count

if current_word:
    print("%s\t%d" % (current_word, current_count))
```

```
import sys
for line in sys.stdin:
    for word in line.strip().split():
        print("%s\t1" % word)
```