```python
import re

line = input().strip()

sylcounts = [5, 7, 5]

words = re.findall('[a-zA-Z]+', line)
originals = line.split()
final = ''

for word, original in zip(words, originals):
    word = word.lower().replace('qu', 'q')
    word = re.sub('y[aoeui]', 'b', word)

    if re.match('.*e$', word) and not re.match('.*[^aoeuiy]le$', word):
        word = word[:-1]

    if re.match('.*es$', word) and not re.match('.*[^aoeuiy][^aoeuiy]es', word):
        word = word[:-2]

    word = re.sub('[aoeuiy]+', 'A', word)

    syllables = word.count('A') or 1

    sylcounts[0] -= syllables
    final += original + ' '

    if sylcounts[0] == 0:
        del sylcounts[0]
        final = final[:-1] + '\n'
        continue
    if sylcounts[0] < 0:
        final = line
        sylcounts = []
        break

if sylcounts:
    final = line

print(final.strip())

```

```python
TESTING = False
inp = '90841947009409037971917182478011970192 68'


def once(values, turnlist):
    values = list(values)
    for ai in reversed(range(len(values))):
        bi = len(values) - ai - 1

        diff = (values[bi] - values[ai]) % 10

        if diff < 5:
            turnlist[ai] += diff
            values[ai] += diff

        if values[ai] > 9:
            if ai > 0:
                values[ai - 1] += values[ai] // 10
            values[ai] = values[ai] % 10

    return values


def is_pal(s):
    return s == s[::-1]


def get(values):
    values = list(map(int, values))

    turns = [0] * len(values)

    while True:
        new = once(values, turns)

        if new == values:
            break

        values = new

    if TESTING:
        print('-----', *values)

    for ai in reversed(range(len(values))):
        bi = len(values) - ai - 1

        if ai == bi:
            pass
        else:
            av = int(values[ai])
            bv = int(values[bi])

            diff = bv - av

            if diff == 5:
                if bi > 0 and turns[bi - 1] > 0:
                    turns[bi] += 5
```

```
                    turns[bi - 1] -= 1
                    values[bi] = (values[bi] + 5) % 10
                else:
                    turns[ai] += 5
                    values[ai] += 5

    return turns, values


if not TESTING:
    inp = input()
else:
    print('input', *inp)

turns, res = get(inp)

if TESTING:
    print()
    print('reslt', *res)
    print('rturn', *res[::-1])
    print(is_pal(res))
    print()
    print('turns', *turns)

print(sum(turns))

```

```python
line = input()

count = 0

min_ = max_ = 0
maxi = mini = 0

for i, let in enumerate(line):
    if let == 'B':
        count += 1
    else:
        count -= 1

    if count < min_:
        min_ = count
        mini = i + 1

    if count > max_:
        max_ = count
        maxi = i + 1

if mini < maxi:
    print(mini + 1, maxi)
elif mini > maxi:
    print(maxi + 1, mini)
else:
    print(mini, mini)

```

```python
from queue import Queue

n = int(input())

tree = {i: [] for i in range(1, n + 1)}

for i in range(n - 1):
    a, b, c = map(int, input().strip().split())

    tree[a].append((b, c))
    tree[b].append((a, c))

keys = list(tree)

bad_paths = {}


def recdel(n, root):
    global bad_paths

    if bad_paths == 'ALL':
        return

    q = Queue()
    q.put((n, root))

    while not q.empty():
        n, root = q.get()

        if n not in tree:
            continue

        if n in bad_paths and bad_paths[n] == root:
            continue

        if root in bad_paths and bad_paths[root] == n:
            bad_paths = 'ALL'  # needs global
            return

        bad_paths[n] = root

        neighbors = tree[n]

        for e, _ in neighbors:
            if e == root:
                continue
            q.put((e, n))


for f in keys:
    if f not in tree:
        continue

    edges = tree[f]
    all_colors = set()
    bad_colors = set()

```

```
    for t, c in edges:
        if c in all_colors:
            bad_colors.add(c)
        else:
            all_colors.add(c)

    for t, c in edges:
        if c in bad_colors:
            recdel(t, f)

all_good = set()
if bad_paths != 'ALL':
    all_good = set(tree) - set(bad_paths)

print(len(all_good))
for i in sorted(all_good):
    print(i)

```

```
1 S = int(input())
2
3 print('%s:' % S)
4
5 for n in range(3, S + 1):
6     if S % n in (0, (n + 1) // 2):
7         print('%s,%s' % ((n + 1) // 2, n // 2))
8
```