```python
#Levenshtein distance
import operator

def Compute(s, t):
    n, m = len(s), len(t)
    d = [[]]
    if (n == 0):
        return m
    if (m == 0):
        return n

    for i in range(n + 1): d.append([i])
    for j in range(m + 1): d[0].append(j)

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            cost = 0 if (t[j - 1] == s[i - 1]) else 1
            d[i].append(min([d[i - 1][j] + 1,
                                 d[i][j - 1] + 1,
                                 d[i - 1][j - 1] + cost]))

    return d[n][m]

data = ["abcd", "abc", "abcde", "ab"]
pattern = "abcde"

letsee = {}
for d in data:
    letsee[d] = Compute(d, pattern)
for j in sorted(letsee.items(), key=operator.itemgetter(1)):
    print(j[0])
```