Zadanie Dostawa – LOGIA 19 SP (2018/19), etap 3

Treść zadania

Fabryka produkuje 3 typy ozdób: *a*, *b* i *c*. Klient określa zamówienie podając kod składający się z liter *a*, *b*, *c*. Kod opisuje, ile ozdób danego rodzaju wchodzi w skład zamówienia. Na przykład kod *baabca*, oznacza, że w skład zamówienia wchodzą trzy ozdoby *a*, dwie *b* i jedna *c*. Zamówienia są realizowane w kolejności zgłoszenia. Jeśli brakuje danego rodzaju ozdoby do zrealizowania zamówienia, to całe zamówienie zostaje anulowane.

Napisz dwuparametrową funkcję **dostawa**, której pierwszym parametrem jest niepusta lista słów reprezentujących zamówienia w kolejności zgłoszeń, a drugim liczba nie mniejsza niż 1 i nie większa niż 250 określająca początkowy zapas każdego rodzaju ozdób. Wynikiem funkcji jest liczba zrealizowanych zamówień.

Przykłady:

Wynikiem **dostawa(["abac", "baaba", "acac", "babc"], 6)** jest **3**

Nie może być zrealizowane trzecie zamówienie, ponieważ zabraknie ozdób typu *a*.

Wynikiem **dostawa(["abc","abaa", "caa", "bbbcc", "cac","abc"], 5)** jest **4**

Nie może być zrealizowane zamówienie trzecie (zabraknie *a*) i szóste.

Omówienie rozwiązania

Rozwiązanie warto podzielić na dwie części: analizowanie poszczególnych słów opisujących konkretne zlecenia oraz przeglądanie listy zamówień i wybieranie tych, które można zrealizować. Zauważmy, że zamówienia są rozpatrywane w kolejności zgłoszenia, a te z nich, których nie można w danej chwili zrealizować odrzucamy. Innymi słowy, nie staramy się zrealizować jak największej liczby zamówień, wyszukując je odpowiednio na liście, lecz przeglądamy listę po kolei.

Dla każdego zamówienia na liście należy policzyć trzy wartości: liczbę ozdób *a*, *b* oraz *c*. Każde z nich wymaga obejrzenia całego kodu zamówienia litera po literze. Gdy w badanym słowie występuje interesująca nas litera zwiększamy odpowiedni licznik. Zamiast pisać trzy osobne funkcje dla trzech typów ozdób, można przygotować jedną funkcję **ile_liter()**. Jej wynikiem będzie lista złożona z trzech liczb, odpowiadających elementom *a*, *b* oraz *c*.

Po przygotowaniu funkcji do szacowania zapasów ozdób niezbędnych do realizacji konkretnego zamówienia, możemy skupić się na zaplanowaniu całej dostawy. Przeglądamy kolejne zamówienia na liście, dla każdego z nich sprawdzając, czy może zostać zrealizowane. Jeśli tak, to zmniejszamy odpowiednio zapasy ozdób i zwiększamy licznik zrealizowanych dostaw.

Rozwiązanie w języku Python

```python
def ile_liter(wyraz):
    wynik = []
    # zerujemy liczniki
    ilea = ileb = ilec = 0
    # zliczamy poszczególne litery w wyrazie
    for litera in wyraz:
        if litera == "a":
            ilea = ilea+1
        if litera == "b":
            ileb = ileb+1
        if litera == "c":
            ilec = ilec+1
    # budujemy listę złożoną z liczby liter a, b i c
    wynik.append(ilea)
    wynik.append(ileb)
    wynik.append(ilec)
    return wynik

def dostawa(lista, zapas):
    ilez = 0
    # ustawiamy początkowe wartości zapasu produktów
    zapas_a = zapas_b = zapas_c = zapas
    # przeglądamy poszczególne zamówienia
    for wyraz in lista:
        # zliczamy produkty w zamówieniu
        w = ile_liter(wyraz)
        # sprawdzamy, czy starczy  zapasu na dane zamówienie
        # jeśli tak, to realizujemy i zwiększamy licznik zamówień
        if (zapas_a >= w[0]) and (zapas_b >= w[1]) and (zapas_c >= w[2]):
            zapas_a = zapas_a - w[0]
            zapas_b = zapas_b - w[1]
            zapas_c = zapas_c - w[2]
            ilez = ilez + 1
    return ilez

```

Testy

Testowanie zaczynamy od przykładów podanych w treści zadania. Następnie dobierając odpowiednio początkowe wartości zapasów oraz listę zamówień staramy się sprawdzić różne możliwe przypadki: gdy można zrealizować wszystkie zlecenia, ani jednego, dokładnie jedno, dwa, trzy itp.

Poniższe testy obejmują proste przypadki, które można bez trudu przeanalizować ręcznie oraz trudniejsze, które obejmują również przypadki wygenerowane losowo.

Wywołanie – Python	**Wynik**
dostawa(["aaa", "bbb", "ccc"], 3)	3
dostawa(["baaacaaa", "bbb", "bbccbbb"], 5)	1
dostawa(["b", "bbb", "bbbbbbb"], 15)	3
dostawa(["baba", "baba", "baba"], 1)	0
dostawa(["baaacaaa", "bbccbbb", "bbb"], 5)	1
dostawa(["aacbacaaccbbbbbcbccbacaababcca", "bcaccaaabcacbbbbacbbabbacaaccc", "abacaabacccabbbbcaaccccbbcbaab", "acabccbacaaabbccaaaabccbbbbbcc", "ccbcbabbabacbacacbcaabcbaacacb", "baabcccabbbbccbaaabcabcabaaccc", "cbabcbcabccabbabccaccaacbbbaaa", "bbacbbcbcbaacbcccabcbabaaaccaa", "babbcaabbaaccacabcabcccbacbabc", "abbcaaabaabbcbbccccbcacacbaacb"], 50)	5
dostawa(["cbacaaccbbbbbcbccbacaa", "ccaaabcacbbbbacbbabbacaac", "bacccabbbbcaaccccb", "acaaabbccaaaabccbbbb", "abbabacbacacbcaabcbaacac", "cabbbbccbaaabcabcabaac", "bcbcabccabbabccaccaacbbb", "cbcbaacbcccabcbab", "baaccacabcabcccbacb", "bcaaabaabbcbbccccbcacacbaac"], 30)	4
dostawa(["acca", "cccab", "cbabacbc", "caabcbca", "baca", "acabba", "bcbc", "abc", "ac", "bcca", "ccbaabc", "bacb", "bbac", "caabcb", "bbcc", "abacbab", "abbca", "ab", "cbbc", "cbbaba", "abc", "ccba", "a", "abcaabc", "cabccaba", "aabcca", "ccb", "abaab", "ccabac", "bcbb", "bbbacc", "c", "acba", "baccaabb", "bacac", "bba", "cccb", "abcaa", "bcbaa", "cabbcc", "accacabbb", "caccbb", "caa", "cbcb", "acccb", "caba", "cccaa", "bcabc", "aaab", "cbab", "bb", "ac", "cacbb", "cabacb", "acca", "bcaaac", "ccab", "babca", "acacbbb", "bab", "cbbaacc", "cbbc", "abbccca", "abbac", "cbbc", "caabcacbb", "c", "abacbca", "caaabcb", "bcbac", "bccbacba", "bcca", "bbcaccaa", "accb", "cbbbcc", "a", "bacaa", "cabca", "cca", "accbc", "bbac", "abaccbc", "bccaa", "bcc", "acbbca", "ccabaabcb", "baca", "bbaacacc", "bac", "acabcac", "baaa", "ababccc", "aabab", "ccbcab", "cacca", "cabaabc", "bccacaab", "cabbb", "acbc", "cabcacba"], 100)	64

dostawa(["acbcba", "bbaaacb", "accbb", "abccabac", "ac", "ccbbaacab", "baba", "baaabc", "baac", "baccab", "bacc", "bbacc", "abcab", "abbb", "bcaa", "c", "cbabab", "bbccbca", "abac", "aacbb", "cacabcab", "cabab", "aabbccac", "accbbba", "caaccb", "abac", "bacacb", "acabb", "bccbca", "bb", "abccb", "aaaccbbc", "bbbc", "ba", "aaa", "caacabcb", "bbcab", "baacba", "acaab", "bcab", "cbbaab", "bbaa", "bcbba", "cba", "cbbcaa", "abaa", "cacc", "bccac", "acacb", "ccbcba", "cbbabacca", "aacabccb", "ccbb", "bcbacaa", "cbcca", "babca", "acccb", "acbbbca", "babaa", "bbcbca", "aaccab", "bab", "bbb", "b", "cbb", "bbbc", "abcc", "bbaba", "ac", "cab", "b", "baa", "accbb", "abaac", "ccc", "cca", "baac", "cbccb", "abbaa", "bbcaabc", "c", "ab", "ba", "cbc", "cbac", "bccba", "aaccbabc", "bbcbaacac", "ccbaa", "acca", "abccaa", "ac", "aaccbbab", "cba", "aaaccb", "aaabcb", "bacc", "acca", "ccbab", "cabaa", "bc", "cac", "acbccbaa", "cbacb", "bbcbac", "cbabccbaa", "bacca", "aba", "bcacabbca", "bccabc", "aacb", "abbcb", "acbcba", "cbac", "aacacb", "cac", "accbb", "cabcba", "a", "cbaaacb", "bbc", "cbabac", "babcbcc", "cbabcc", "cbaccab", "ccbbabaca", "abbbac", "bcca", "cabbba", "ca", "cbacbc", "aacbbacc", "cac", "bca", "caacbb", "bbacacc", "ccba", "abbcabc", "baba", "bacba", "bcbaacb", "acbbba", "aaa", "bbbacac", "acb", "cbcaacb", "cabcaac", "aabcbbcc", "bcbcacbaa", "baaccbc", "bcca", "abcb", "cac", "bcbac", "bcaabc", "babc", "aaccb", "baacba", "accabbcb", "cb", "ca", "ccbbaaab", "caccbbaab", "bbca", "bcacaabc", "cabbcb", "abca", "cabcba", "cc", "caabbbac", "abbaa", "caccb", "aa", "baca", "aacbbcb", "cbbaacba", "bac", "caabbbca", "bbabccca", "aabccab", "bcacbbc", "cbcaaa", "ccaab", "bcba", "cba", "b", "aabb", "abcaacb", "baccabba", "bcac", "bb", "acac", "bbacbcc", "cb", "accbb", "ccbbcbaaa", "babca", "aaccc", "cbbcac", "aabccab"], 150)	96
dostawa(["cbab", "ccaaca", "acb", "babbaccc", "abc", "cbabca", "ccacbb", "acabcb", "bacbaacc", "ccbbca", "bca", "ccbcaa", "ccabc", "cbbaabc", "bcaab", "caabab", "babac", "aca", "bacab", "aabacbcb", "acbc", "bac", "ccb", "bac", "caac", "ba", "ccaab", "ccba", "bac", "acbcab", "ba", "acbcac", "cbac", "aac", "abcbbc", "aacc", "aaccbca", "ccc", "abcabac", "acbbaca", "cbcca", "ca", "ac", "acccb", "aaabbcc", "aaa", "cabbac", "baab", "accbcb", "ccb", "bccaca", "cbcac", "ab", "cab", "bc", "cc", "b", "cbaacbc", "bba", "cbacbcbaa", "aa", "bcabaac", "caaaccb", "aca", "baca", "acbacb", "bacaa", "ccacbbaab", "abcba", "bcb", "abccb", "baaabc", "cabc", "cabccba", "bc", "cacbaacb", "abbcb", "cbca", "abcca", "ccacb", "bcac", "abbcaa", "cabcba", "baccb", "bbca", "abbcaa", "cabcbcab", "ac", "cbccbb", "ccb", "acabbbac", "cbcaa", "caabcbb", "babca", "bab", "cbaccbbaa", "cba", "cabbb", "ababacccb", "aabcbcba", "bcabcba", "bcbacab", "aabc", "cbaaacc", "cccab", "aba", "aaaccb", "bcbba", "cabbaba", "bb", "caa", "bab", "bacca", "bcbbc", "ccbbbc", "babc", "caacac", "cabaa", "bccab", "bcba", "bab", "ccbaac", "babba", "acacbb", "baabc", "bc", "cabbaac", "ca", "babac", "bcca", "caca", "abacca", "caabb", "cac", "bb", "acbb", "b", "bacbbcaa", "bcbc", "cbabbc", "cb", "bcbabaca", "ccabb", "bcbba", "cabacbba", "babba", "abccbb", "abcba", "baabcbca", "cc", "b", "aa", "cbbbca", "accbb", "accaabb", "acbccaa", "bbcacaa", "aabccab", "ccbca", "bacb", "cbacba", "aca", "ac", "cc", "aaacbcb", "ababccab", "bcb", "ccbac", "caca", "a", "bbcaba", "aa", "ccaacbb", "aacabb", "abcccaabb", "cbacaa", "babcabc", "bccaa", "aacb", "cbcabbaca", "baacc", "cbcca", "bcabaa", "abc", "cbcbaa", "cccabaab", "cacbcab", "abbaca", "abaac", "abacba", "abac", "baabc", "bcaaca", "bc", "cbbaa", "bba", "cbaacbbc", "accbbb", "aacbaccb", "abcabac"], 250)	156