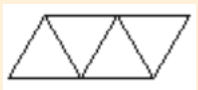
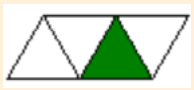
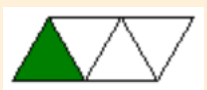
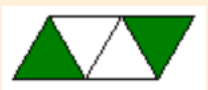
Zadanie Szyfr trójkątny – LOGIA 23 (2022/23), etap 2

Treść zadania

Adaś postanowił kolejne cyfry systemu dziesiętnego przedstawiać za pomocą czterech bitów (cyfr 0 lub 1). Kolejne cyfry od 0 do 9 przedstawione w ten sposób to:

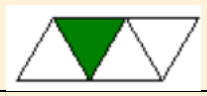
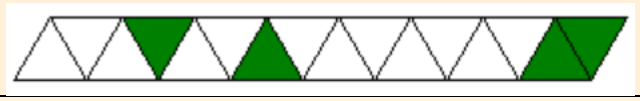
0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001.

Kolejny etap szyfru Adasia polega na przedstawieniu każdej cyfry od 0 do 9 w postaci czterech trójkątów, zamalowanych kolorem białym lub zielonym, w zależności od tego, czy dany trójkąt reprezentuje 0 lub 1 w cyfrze dziesiętnej przedstawionej na czterech bitach.

				
cyfra 0 – 0000	cyfra 1 – 0001	cyfra 2 – 0010	cyfra 3 – 0011	cyfra 4 – 0100
				
cyfra 5 – 0101	cyfra 6 – 0110	cyfra 7 – 0111	cyfra 8 – 1000	cyfra 9 – 1001

Napisz funkcję **pasek(n)**, po wywołaniu której powstanie rysunek paska powstałego po zaszyfrowaniu liczby całkowitej z zakresu od 0 do 10^{10}. Kolejne cyfry liczby podanej jako parametr przedstawione są na rysunku w postaci czterech trójkątów reprezentujących każdą z cyfr. Długość boku trójkąta wynosi 26. Rysunek powinien być jednakowo odległy od lewej i prawej strony ekranu.

Przykłady:

	
`pasek(4)`	`pasek(95)`
	
`pasek(1203)`	

Omówienie rozwiązania

Można zauważyć, że cyfry systemu dziesiętnego są przedstawione za pomocą czterech bitów. Warto napisać funkcję zamiany cyfry w systemie dziesiętnym na liczbę binarną. Wynik funkcji – liczba binarna musi zawsze mieć 4 cyfry niezależnie od jej wartości.

```
def szyfruj(cyfra):
    dw =''
    for i in range(4):
        dw = str(cyfra % 2)+ dw
        cyfra = cyfra // 2
    return dw
```

Analizując zapis binarny cyfry dziesiętnej rysujemy kolejne trójkąty zamalowane kolorem białym w przypadku zera i kolorem zielonym w przypadku cyfry jeden.

```
def jedna(bok, cyfra):
    d = szyfruj(cyfra)
    tr(bok, d[0]); fd(bok); lt(60)
    tr(bok, d[1]); rt(60)
    tr(bok, d[2]); fd(bok); lt(60)
    tr(bok, d[3]); rt(60)
```

Wykorzystujemy funkcję pomocniczą rysowania trójkąta zamalowanego kolorem podanym jako drugi parametr (pierwszym parametrem jest długość boku trójkąta).

```
def tr(bok, kolor):
    if kolor == '0':
        fillcolor('white')
    else:
        fillcolor('green')
    begin_fill()
    for i in range(3):
        fd(bok); lt(120)
    end_fill()
```

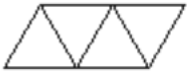
W funkcji głównej **pasek(liczba)** zaczynamy od zamiany liczby na napis, co ułatwi przeglądanie podanej jako parametr liczby. Następnie deklarujemy zmienną **bok** i ustalamy zgodnie z treścią zadania jej wartość na 26.

Ustawiamy żółwia w takim miejscu, aby po narysowaniu paska rysunek był jednakowo odległy od lewej i prawej strony ekranu. Zauważmy, że szerokość rysunku to *2 * liczba cyfr liczby podanej jako parametr * długość boku trójkąta + połowa długości boku trójkąta*.

```
def pasek(liczba):
    liczba = str(liczba)
    bok = 26
    pu();bk(bok * len(liczba) + bok / 4);pd()
    for i in range(len(liczba)):
        jedna(bok,int(liczba[i]))
```

Testy

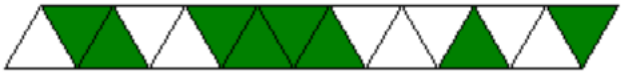
Sprawdzamy czy rysunek paska jest prawidłowy dla podanych przykładów z treści zadania. Podczas konkursu rozwiązanie było testowane dla poniższych danych.



pasek(0)

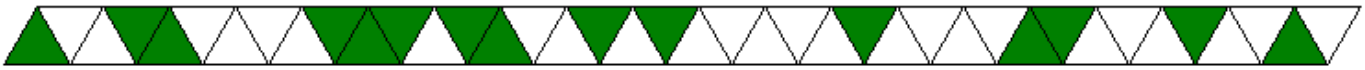
pasek(12)

pasek(345)



pasek(6789)

pasek(888888888)



pasek(9876504312)

Zadanie Kodowanie – LOGIA 23 (2022/23), etap 2

Treść zadania

Ignacy koduje wiadomość. W tym celu wymyślił własną metodę. Polega ona na wypisaniu najpierw wszystkich samogłosek występujących w wiadomości w kolejności alfabetycznej, następnie spółgłosek też w kolejności alfabetycznej. Jeśli jakaś litera występuje więcej niż jeden raz, to Ignacy wypisuje ją raz i po niej liczbę jej wystąpień. Napisz program kodujący wiadomość zgodnie z metodą Ignacego.

Wejście

Liczba całkowita **n** z zakresu od 1 do 100 określająca liczbę wierszy kodowanej wiadomości.

W kolejnych **n** wierszach znajduje się tekst składający się z małych liter alfabetu łacińskiego (samogłoski alfabetu łacińskiego to: a, e, i, o, u, y). Każdy wiersz tekstu składa się z co najmniej jednej litery i jest nie dłuższy niż 100 liter.

Wyjście

n wierszy tekstu – zakodowana wiadomość

Przykłady:

Wejście	**2** **alabama** **kwiatuszek**	**1** **ogrodzenie**	**3** **akwarela** **butonierka** **zryw**
Wyjście	**a4blm** **aeiuk2stwz**	**e2io2dgnrz**	**a3eklrw** **aeioubknrt** **yrwz**

Omówienie rozwiązania

Zacznijmy od alfabetu łacińskiego – składa się on z 26 liter, które są na klawiaturze komputera i nie trzeba znać kolejności liter alfabetu na pamięć. Wystarczy go wygenerować wykorzystując funkcję `chr(kod)`, zamieniającą kod ASCII na odpowiednią literę alfabetu.

```
1 alfabet = ''
2 for i in range(26):
3     alfabet += chr(97 + i)
4 print(alfabet)
```

Po uruchomieniu tego programu zostanie wypisany alfabet złożony z małych liter (kody ASCII małych liter rozpoczynają się od 97).

```
abcdefghijklmnopqrstuvwxyz
```

Zanim zaczniemy rozwiązywać zadanie omówimy parametry funkcji `print()`. Zazwyczaj podaje się tylko dane do wypisania na ekranie – tekst, liczbę, listę itp. Funkcja `print()` posiada jeszcze dwa parametry opcjonalne: `sep = 'wartość'` oraz `end = 'wartość'`. Parametr `sep` powoduje, że

podczas wypisywania kilku wartości rozdzielonych przecinkiem, domyślny separator, jakim jest spacja, zostanie zamieniony na podany. Na przykład:

```
print('Ala', 'ma', 'kota', sep = '#')
```

zostanie wypisany napis **Ala#ma#kota**

Parametr `end` spowoduje zamianę znaku końca wiersza na podany, czyli po wypisaniu wartości kursor nie przejdzie do następnej linii, tylko zostanie wypisana podana wartość.

Na przykład polecenie:

```
print('Ala', 'ma', 'kota', sep = '#', end = '$')
print('drugi')
```

wypisze na ekranie trzy napisy oddzielone znakiem `'#'` zamiast spacji, a kursor nie zostanie przeniesiony do następnej linii, ponieważ zostanie zastąpiony znakiem `'$'`. Na ekranie pojawi się napis:

Ala#ma#kota$drugi

Metodę zliczania występujących samogłosek i spółgłosek można wybrać, sugerując się oczekiwanym wynikiem. Należy wypisywać w kolejności alfabetycznej najpierw samogłoski, potem spółgłoski. Dlatego warto zliczać kolejno samogłoski i wypisywać je, jeśli dana samogłoska wystąpiła w badanym słowie, oraz liczbę jej wystąpień, jeśli jest większa niż jeden. Podczas wypisywania separator zastępujemy pustym znakiem `''` (2 apostrofy) i tak samo robimy ze znakiem końca wiersza.

```
1  sam = 'aeiouy'
2  w = input()
3  for j in range(len(sam)):
4      ile = w.count(sam[j])
5      if ile == 1:
6          print(sam[j], end='')
7      elif ile > 1:
8          print(sam[j], ile, sep = '', end = '')
```

Na przykład dla wczytanego napisu **aeaeaeooouyo** zostanie wypisany wynik **a3e3o4uy**.

Zliczanie spółgłosek jest identyczne, tylko wartość zmiennej pomocniczej, to wszystkie spółgłoski w kolejności alfabetycznej.

Program na początku wczytuje liczbę całkowitą **n** określającą liczbę wierszy kodowania. Potem są zadeklarowane zmienne przechowujące samogłoski i spółgłoski w kolejności alfabetycznej. W pętli `for` powtarzamy **n** razy wczytywanie tekstu. Dla każdego tekstu zliczamy kolejno samogłoski, jeśli dana samogłoska występuje raz, to wypisujemy ją, jeśli więcej razy, to wypisujemy ją i liczbę jej wystąpień. Po samogłoskach zliczamy spółgłoski. Cały czas wszystkie wartości wypisujemy znak po znaku, pusty wiersz `print()` jest wstawiony dopiero po przeanalizowaniu całego wczytanego napisu.

```
n = int(input())
sam = 'aeiouy'
sp = 'bcdfghjklmnpqrstvwxz'
for i in range(n):
    w = input()
    for j in range(len(sam)):
        ile = w.count(sam[j])
        if ile == 1:
            print(sam[j], end='')
        elif ile > 1:
            print(sam[j], ile, sep = '', end = '')
    for j in range(len(sp)):
        ile = w.count(sp[j])
        if ile == 1:
            print(sp[j], end='')
        elif ile > 1:
            print(sp[j], ile, sep = '', end = '')
    print()
```

Testy

Testujemy działanie programu dla przykładów z treści zadania. Warto również samemu wymyśleć i uruchomić testy zawierające dłuższe teksty.

Na przykład dla testu:

```
3
elementarz
serdeczniejimilej
abrakadabrahokuspokus
```

wynikiem będzie:

```
ae3lmnrtz
e4i3cdj2lmnrsz
a5o2u2b2dhk3pr2s2
```

Dla testu:

```
2
mhkajircduodjffichvnsgpzheessakrwrpbmjemrfvbjdqzbu
oovigzxesglopeufwdhzcpnacozghehxujxlygcszlelgtdvxzkdszgx
```

wynik będzie następujący:

```
a2e3i2ou2b3c2d3f3gh3j4k2m3np2qr4s3v2wz2
ae4io4u2yc3d3fg6h3jkl4np2s3tv2wx5z6
```

Zadanie Ogrodzenie – LOGIA 23 (2022/23), etap 2

Treść zadania

Zosia zaprojektowała ogrodzenie dookoła swojego ogrodu. Składa się ono z paneli różnej wysokości. Zapomniała jednak o dwóch furtkach, które planowała umieścić w ogrodzeniu. Zastanawia się teraz, które panele opłaca się zastąpić furtkami. Chce wymienić dwa panele na furtki, ale tak, by furtki nie sąsiadowały ze sobą oraz koszt ogrodzenia był jak najniższy. Koszt panelu jest równy jego wysokości, a furtki Zosia otrzymała w prezencie za darmo. Napisz program, który obliczy koszt ogrodzenia po wymianie paneli na furtki.

Wejście

Pierwszy wiersz wejścia zawiera jedną liczbę naturalną **n** oznaczającą liczbę paneli w ogrodzeniu, $5 \le n \le 1000$.

Drugi wiersz zawiera **n** liczb naturalnych z przedziału [1; 10000] rozdzielonych spacją, opisujących wysokości kolejnych paneli w ogrodzeniu.

Wyjście

Liczba naturalna – minimalny koszt ogrodzenia.

Przykłady:

Wejście	**10** **5 5 6 7 3 8 5 3 5 3**	**5** **5 5 5 7 8**	**5** **7 5 5 5 8**
Wyjście	**35**	**17**	**17**
Komentarz	furtki zastąpiły panele o wysokości 7 i 8 koszt ogrodzenia: 5 + 5 + 6 + 0 + 3 + 0 + 5 + 3 + 5 + 3 = 35	furtki zastąpiły panele o wysokościach 5 i 8 uwaga: panele o wysokościach 7 i 8 sąsiadują ze sobą!	furtki zastąpiły panele o wysokościach 5 i 8 uwaga: panele o wysokościach 7 i 8 sąsiadują ze sobą!

Omówienie rozwiązania

Na pierwszy rzut oka rozwiązanie polega na znalezieniu dwóch maksymalnych wartości w podanym ciągu. Niestety jest jeden dodatkowy warunek – poszukiwane wartości nie mogą ze sobą sąsiadować. Dotyczy to również pierwszego i ostatniego elementu. Czy uwzględniając dodatkowy warunek wystarczy poszukać trzech maksymalnych wartości? Otóż nie. Poniższy przykład to demonstruje:

```
1 100 10000 101 2
```

Wyszukując trzy największe wartości otrzymamy 10000, 101 i 100. Po uwzględnieniu dodatkowego warunku wybieramy liczby 100 i 101 (nie sąsiadują ze sobą) i w wyniku otrzymujemy 10003 (1 + 10000 + 2). Nie jest to poprawna odpowiedź, ponieważ bardziej opłaca się usunąć z ciągu

10000 i 2. Wtedy wynik to 203 (1 + 100 + 101). Z tej prostej obserwacji wynika, że do ustalenia poprawnego wyniku poszukujemy czterech największych wartości, spośród których wybierzemy dwie niesąsiadujące ze sobą o maksymalnej sumie. Żeby prawidłowo ustalić wynik musimy oprócz liczby zapamiętywać również jej pozycję w ogrodzeniu.

Zacznijmy od najwolniejszego rozwiązania. Zbadajmy wszystkie pary liczb niesąsiadujących ze sobą w poszukiwaniu maksymalnej sumy dwóch liczb.

```
maksymalna ← 0
dla i od 0 do n - 3 wykonuj
    dla j od i + 2 do n - 1 wykonuj
        jeżeli nie (i = 0 oraz j = n - 1) oraz p[i] + p[j] > maksymalna to
            maksymalna = p[i] + p[j]
```

Rozwiązanie w języku Python

```
 1 n = int(input())
 2 lista = input().split()
 3 p = []
 4 suma = 0
 5 for i in range(n):
 6     p.append(int(lista[i]))
 7     suma = suma + int(lista[i])
 8 maksymalna = 0
 9 for i in range(n - 2):
10     for j in range(i + 2, n):
11         if not (i==0 and j==n-1) and p[i] + p[j] > maksymalna:
12             maksymalna = p[i] + p[j]
13 print(suma-maksymalna)
```

Powyższe rozwiązanie jest proste do implementacji i ma złożoność czasową wystarczającą do danych z treści zadania (1000 liczb).

Szybsze rozwiązanie sprowadza się do znalezienia czterech największych wartości.

```
 1 maks4 = [[0, 0], [0, 0], [0, 0], [0, 0]]
 2 for i in range(n):
 3     if p[i] > maks4[0][0]:
 4         maks4.insert(0, [p[i], i])
 5         maks4.pop()
 6     elif p[i] > maks4[1][0]:
 7         maks4.insert(1, [p[i], i])
 8         maks4.pop()
 9     elif p[i] > maks4[2][0]:
10         maks4.insert(2, [p[i], i])
11         maks4.pop()
12     elif p[i] > maks4[3][0]:
13         maks4.insert(3, [p[i], i])
14         maks4.pop()
```

Zauważmy, że tylko raz przeglądaliśmy zbiór danych.

Po ustaleniu czterech największych wartości (lista maks4) można sprawdzić, które liczby będą stanowiły rozwiązanie. Do rozpatrzenia jest sześć par (0, 1), (0, 2), (0, 3), (1, 2), (1, 3) i (2, 3).

```
maksymalna = 0
for i in range(3):
    for j in range(i + 1, 4):
      if (not (maks4[i][1]==0 and maks4[j][1]==n-1)) and
         (not (maks4[i][1]==n-1 and maks4[j][1]==0)) and
         abs(maks4[i][1] - maks4[j][1]) > 1 and
         maks4[i][0] + maks4[j][0] > maksymalna:
            maksymalna = maks4[i][0] + maks4[j][0]
```

Poniżej kod całego programu.

```
n = int(input())
lista = input().split()
p = []
suma = 0
for i in range(n):
    p.append(int(lista[i]))
    suma = suma + int(lista[i])
maks4 = [[0, 0], [0, 0], [0, 0], [0, 0]]
for i in range(n):
    if p[i] > maks4[0][0]:
        maks4.insert(0, [p[i], i])
        maks4.pop()
    elif p[i] > maks4[1][0]:
        maks4.insert(1, [p[i], i])
        maks4.pop()
    elif p[i] > maks4[2][0]:
        maks4.insert(2, [p[i], i])
        maks4.pop()
    elif p[i] > maks4[3][0]:
        maks4.insert(3, [p[i], i])
        maks4.pop()
maksymalna = 0
for i in range(3):
    for j in range(i + 1, 4):
        if (not (maks4[i][1]==0 and maks4[j][1]==n-1)) and
           (not (maks4[i][1]==n-1 and maks4[j][1]==0)) and
           abs(maks4[i][1] - maks4[j][1]) > 1 and
           maks4[i][0] + maks4[j][0] > maksymalna:
               maksymalna = maks4[i][0] + maks4[j][0]
print(suma-maksymalna)
```

Testy

Rozwiązanie należy przetestować najpierw na przykładach z treści zadania, następnie na większych danych. Podczas konkursu zdanie było testowane na następujących grupach testów.

Grupa testów	Test	Wynik	Uwagi
I	6 1 2 6 3 5 4	10	dwie największe wartości (brak sąsiedztwa, bez brzegu)
	6 40 50 30 60 20 10	100	
	6 100 500 200 300 600 400	1000	
II	6 2 99 100 1 98 1	103	sąsiedztwo dwie największe, bez brzegu
	11 7 7 123 122 8 8 8 9 9 99 9	187	
	20 2 456 1000 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1	481	
III	6 100 1 98 2 1 99	103	sąsiedztwo dwie największe na brzegu
	60 500 1 1 1 1 1 1 1 1 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 498 1 1 1 1 1 1 1 1 1 1 1 1 499	566	
	100 298 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 300 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 299	824	
IV	6 1 99 10000 98 2 1	199	trzy największe obok siebie, bez brzegu
	100 8 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 90 110 100 3 4 5 6 7 8 79 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2	705	
	1000 8 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 118 7 6 5 4 3 2 3 4 5 6 7 8 980 10000 999 3 4 5 6 7 8 3 4 5 6 7 8 3 4 5 6 7 8 3 4 5 6 7 8 3 4 5 6 7 8 118 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7	8051	

	8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 118 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 118 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 118 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 118 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 118 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5		
V	6 10000 98 2 1 1 99	199	trzy największe obok siebie, dwie na brzegu
	100 99 3 4 5 6 7 8 79 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 8 100 110	723	
	1000 999 3 4 5 6 7 8 3 4 5 6 7 8 3 4 5 6 7 8 3 4 5 6 7 8 3 4 5 6 7 8 118 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 118 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3	8051	

	4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 118 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 118 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 118 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 4 3 2 3 4 5 6 7 8 9 118 7 6 5 4 3 2 3 4 5 6 2 3 4 5 6 7 8 9 8 7 6 8 7 6 5 8 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 3 4 5 6 7 8 9 118 7 6 5 4 3 2 3 4 5 6 7 8 980 10000		

Zadanie Pamięć absolutna – LOGIA 23 (2022/23), etap 2

Treść zadania

Wojtek jest miłośnikiem gry Memory. Rekwizytami w grze Wojtka są karty oznaczone liczbami od **1** do **n**. Każda liczba występuje na dwóch kartach. Wojtek tasuje karty i rozkłada je kolejno wierzchem do góry (zakrytą stroną). Ruch polega na odsłonięciu dowolnej pary z zakrytych kart. Jeżeli odkryta para zawiera różne liczby, to Wojtek z powrotem je zakrywa. Jeżeli liczby są takie same, to usuwa je z gry. Gra kończy się po usunięciu wszystkich kart. Wojtek przyjął następną strategię gry: karty odkrywa od końca i zapamiętuje położenie już odkrytych kart. Napisz program, który policzy liczbę ruchów potrzebną do zakończenia gry przy zastosowaniu strategii Wojtka.

Przykład:

Ciąg kart: 1 5 3 4 5 2 3 4 2 1

Ruch 1: Wojtek odkrywa karty z 1 i 2 – zapamiętuje położenie i zakrywa karty;

Ruch 2: Wojtek odkrywa karty z 4 i 3 – zapamiętuje położenie i zakrywa karty;

Ruch 3: Wojtek odkrywa kartę z 2 oraz wcześniej zapamiętaną kartę z 2 – usuwa parę;

Ruch 4: Wojtek odkrywa karty z 5 i 4 – 4 już była, ale została odsłonięta jako druga;

Ruch 5: Wojtek odkrywa i usuwa karty z 4;

Ruch 6: Wojtek odkrywa kartę z 3 oraz wcześniej zapamiętaną kartę z 3 – usuwa parę;

Ruch 7: Wojtek odkrywa kartę z 5 oraz wcześniej zapamiętaną kartę z 5 – usuwa parę;

Ruch 8: Wojtek odkrywa i usuwa ostatnią parę.

Wejście

Pierwszy wiersz wejścia zawiera liczbę naturalną **n**, $2 \le n \le 1000$

Drugi wiersz wejścia zawiera **2*n** liczb naturalnych z zakresu **[1, n]** oddzielonych spacją, każda liczba występuje dokładnie dwa razy.

Wyjście

Liczba naturalna określająca liczbę ruchów wykonanych w grze zgodnie ze strategią Wojtka.

Przykłady:

Wejście	**5** **1 5 3 4 5 2 3 4 2 1**	**3** **1 1 2 2 3 3**	**6** **6 6 5 4 3 5 2 4 1 3 2 1**
Wyjście	**8**	**3**	**10**

Omówienie rozwiązania

Gra Memory jest popularną grą towarzyską. Rozgrywana jest przez co najmniej dwie osoby. Wariant opisany w tym zadaniu jest raczej rodzajem pasjansa. Anglojęzyczna Wikipedia pod hasłem

Concentration[1] podaje link do artykułu w którym wykazano, że na optymalne odsłanianie przez jedną osobę kart potrzebne jest średnio **1,61 * n** ruchów dla dużych **n**. Historia gry opisana jest m.in. w niemieckojęzycznej Wikipedii.

Chcąc obliczyć minimalną liczbę ruchów dla wylosowanego układu kart (w zadaniu reprezentowanego przez liczby) musimy wybrać sposób zapamiętywania odkrytych kart. Wykorzystamy do tego tablicę wartości logicznych (*odkryta*). Wprowadzimy także trzy pomocnicze zmienne, do zliczania ruchów (*ruchy*), do oznaczenia, która karta z pary jest w danym momencie odsłaniana (*karta*) oraz do zapamiętania wartości pierwszej odkrywanej karty z pary (*poprzednia*) – dla przypadku, gdy odkrywamy dwie takie same sąsiednie karty w jednym ruchu i je zbieramy.

Algorytm polega na przejrzeniu **2 * n** liczb od ostatniej wczytanej liczby do pierwszej. Jeśli dana liczba jest pierwszą z pary, to sprawdzamy, czy wystąpiła już wcześniej. Jeśli tak, to jako drugą do pary bierzemy ją, powiększamy więc licznik ruchów. W przeciwnym przypadku (nie wystąpiła wcześniej) zapamiętujemy jej wartość oraz oznaczamy jako odkrytą i przechodzimy do drugiej liczby z pary. Dla drugiej liczby zawsze powiększamy licznik ruchów. Następnie sprawdzamy, czy wystąpiła już wcześniej i jest różna od poprzedniej karty. Jeśli tak, to kolejny ruch polega na zebraniu tej pary, a więc powiększamy licznik ruchów. Jeśli nie wystąpiła wcześniej oznaczamy ją jako odkrytą i przechodzimy do kolejnej liczby jako pierwszej z pary w kolejnym ruchu.

Zapis algorytmu w pseudokodzie może być następujący:

```
wczytaj n
wczytaj tablicę K                      # 2*n liczb reprezentujących karty
ruchy ← 0                              # inicjalizacja licznika ruchów
dla j = 0..n-1 odkryta[j] ← fałsz      # inicjalizacja tablicy z odkrytymi kartami
karta ← 1                              # która karta z pary jest odkrywana
dla każdego z od K[2*n-1] do K[0]      # wykonuj dla kolejnych kart od końca
  jeśli karta = 1 to
        jeśli odkryta[z] to ruchy ← ruchy + 1
        w przeciwnym przypadku
              poprzednia ← z
              karta ← 2
  w przeciwnym przypadku
        ruchy ← ruchy + 1
jeśli odkryta[z] oraz poprzednia ≠ z to ruchy ← ruchy + 1
karta ← 1
  odkryta[z] ← prawda
wypisz(ruchy)
```

Warto zwrócić uwagę, że dla każdej oglądanej liczby możemy ustawić, że jest ona odkryta (nie ma przy tym znaczenia, czy była ona już wcześniej odsłonięta).

[1] https://en.wikipedia.org/wiki/Concentration_(card_game)

Rozwiązanie w języku Python

```
n = int(input())
K = [int(i) for i in input().split()]
odkryta = [False] * (n+1)
ruchy = 0
karta = 1
for z in K[::-1]:       #przeglądamy tablicę od ostatniego elementu
    if karta == 1:
        if odkryta[z]: ruchy += 1
        else:
            poprzednia = z
            karta = 2
    else:
        ruchy += 1
        if odkryta[z] and poprzednia != z: ruchy += 1
        karta = 1
    odkryta[z] = True
print(ruchy)
```

Testy

Podczas konkursu zdanie było testowane na następujących grupach testów.

Grupa testów	Test	Wynik
I	3 2 3 1 1 2 3	4
	2 1 2 1 2	3
II	5 5 4 3 3 1 2 2 4 1 5	7
	6 2 5 3 2 1 1 3 5 6 4 4 6	9
III	20 14 13 15 2 20 19 10 3 17 10 16 4 6 4 7 7 14 12 12 19 18 8 1 9 20 9 16 8 3 5 15 11 5 6 11 18 1 2 13 17	32
	25 23 25 12 4 7 17 8 6 6 18 13 18 5 3 11 10 14 12 16 2 5 15 20 1 24 22 16 24 9 15 25 20 17 21 19 8 10 7 21 3 22 23 13 2 11 4 9 19 14 1	39
IV	100 72 76 58 53 68 50 54 44 5 78 50 87 15 10 88 93 65 56 35 80 89 59 37 6 42 62 55 96 55 100 74 95 52 64 10 15 46 26 47 40 13 97 21 29 8 6 59 30 83 38 11 11 93 86 8 19 84 96 58 97 67 1 98 81 63 53 92 69 22 85 81 90 22 82 66 17 41 67 54 3 39 2 49 76 18 84 77 98 31 41 12 14 47 94 86 62 48 32 77 80 60 9 46 33 23 36 7 66 20 73 32 27 51 83 88 37 7 28 52 44 91 18 33 17 9 23 99 40 38 2 28 68 39 61 64 89 24 45 65 57 48 3 78 16 63 91 57 61 31 25 43 36 75 87 4 29 43 5 34 73 49 70 13	162

	72 71 94 30 82 24 26 100 90 79 92 75 20 69 85 14 12 21 45 74 42 60 19 56 71 25 16 27 99 70 34 51 1 79 35 95 4	
	97 75 28 49 19 11 16 25 20 41 76 45 74 50 80 23 31 48 92 79 93 69 60 64 51 95 51 54 91 32 84 35 81 55 78 93 81 48 89 47 19 83 5 66 43 85 97 6 52 46 38 35 50 72 87 92 5 33 9 75 24 42 70 67 77 30 33 62 12 64 38 15 15 37 90 10 3 30 68 71 65 85 68 47 17 7 89 20 94 84 72 56 58 61 34 91 86 69 59 14 61 41 43 9 53 74 13 11 10 90 82 37 22 29 7 25 26 2 88 24 32 17 96 8 3 27 8 36 76 13 40 27 71 97 40 87 65 86 28 63 36 52 88 59 66 29 39 82 57 73 83 53 94 18 4 46 26 54 1 21 34 44 45 21 39 16 57 80 96 1 67 6 95 58 2 60 70 44 78 12 31 22 73 42 79 4 49 18 23 77 14 62 56 63 55	155
V	Test a poniżej	1623
	Test b poniżej	1597

Test a)

1000

1000 46 940 346 498 251 509 514 640 344 336 952 832 901 63 644 415 434 835 40 543 919 770 91 829 189 790 671 186 284 125 123 136 638 756 400 254 505 568 696 102 607 274 676 558 634 175 49 381 218 999 331 663 870 534 753 904 518 684 389 282 312 435 779 154 122 695 969 200 607 633 442 677 432 860 993 104 672 621 560 503 371 887 469 728 579 47 60 308 613 587 900 271 266 366 132 859 157 904 648 235 501 43 296 242 965 56 208 90 993 931 522 78 455 124 815 711 577 922 867 507 865 483 197 359 178 232 848 268 619 55 576 849 316 970 727 174 74 633 584 772 874 933 227 927 998 292 865 570 946 902 302 780 877 322 314 183 484 85 34 239 460 439 604 528 424 542 416 334 837 881 255 306 659 448 239 864 355 5 489 311 248 547 598 447 62 792 403 685 710 666 921 797 744 625 521 842 180 638 391 505 918 298 942 647 241 897 610 630 125 687 360 725 586 69 363 863 692 697 712 351 390 916 810 658 42 106 467 523 582 734 889 841 537 366 750 342 958 254 226 795 134 38 210 415 275 783 930 45 826 114 449 867 482 535 390 573 978 647 461 992 407 227 224 157 661 183 892 702 97 12 651 385 375 501 170 790 463 240 83 943 226 496 998 991 222 339 398 953 292 285 473 92 856 694 670 46 632 553 399 59 622 176 608 51 479 908 961 33 606 100 577 517 317 70 392 691 358 668 983 719 775 394 794 52 251 294 113 624 151 287 526 872 538 168 491 678 989 628 784 210 737 199 728 504 801 883 169 569 285 410 713 116 617 347 695 563 799 740 149 282 162 763 365 357 198 497 348 333 270 688 491 172 699 279 341 824 548 88 24 769 448 712 503 939 488 149 710 115 421 511 592 946 671 899 590 839 506 857 793 542 109 151 428 28 288 15 34 187 297 760 345 907 855 293 112 281 305 660 614 144 268 947 721 329 471 114 686 326 842 418 37 752 28 431 539 9 862 809 841 458 573 55 545 160 731 800 425 202 123 165 600 622 373 480 830 208 142 855 63 22 296 58 798 425 16 86 379 395 813 807 345 552 470 118 962 54 460 156 596 378 861 277 59 603 304 627 452 580 832 657 706 310 854 89 926 934 562 276 185 512 967 720 93 847 609 927 778 401 436 446 45 337 836 313 562 571 784 445 367 670 747 141 220 554 464 814 434 786 417 549 864 818 473 126 206 940 910 71 112 387 681 143 203 697 319 14 885 926 207 446 742 180 925 163 177 956 95 400 544 625 280 193 219 850 513 759 532 869 963 979 624 895 132 722 459 986 234 327 456 238 90 64 388 274 110 177 32 419 80 215 178 374 454 949 587 393 301 758 256 994 767 725 3 908 636 968 10 724 609 104 669 397 315 561 507 602 585 223 880 377 810 174 641 233 653 186 708 225 196 494 211 964 228 39 256 623 241 117 796 663 525 618 856 515 259 727 474 899 472 17 85 67 330 181 500 892 35 335 769 406 70 975 200 774 944 304 564 187 120 171 829 643 752 468 875 309 257

871 873 567 492 987 301 429 821 365 873 565 356 26 221 601 379 584 968 428 99 770 581 914 461
417 14 852 107 948 871 468 594 262 244 172 693 436 453 525 813 399 409 54 971 987 935 809 676
302 22 190 142 18 683 579 50 467 295 527 100 225 620 517 965 694 533 11 398 976 20 920 128 831
575 558 772 971 230 737 907 851 582 654 166 360 929 481 356 40 396 41 99 843 845 717 211 347
972 567 816 950 138 683 982 451 745 541 537 886 342 652 1 597 964 822 948 653 392 458 870 61
331 137 248 974 133 516 539 626 782 980 140 909 435 707 718 741 181 887 190 135 405 402 255 237
739 353 164 662 66 924 962 668 290 105 807 65 644 550 639 246 205 322 307 768 198 374 75 746
578 787 662 523 395 376 103 941 767 266 368 362 818 673 876 880 797 116 529 140 385 742 440 84
536 222 526 598 243 937 824 891 155 182 879 959 561 760 931 951 840 909 377 324 533 616 718 404
819 194 298 833 966 326 701 74 766 75 297 761 294 733 736 589 250 253 884 822 827 611 249 893
949 271 762 135 756 632 519 929 709 995 597 486 945 438 764 146 821 859 240 628 321 546 990 817
789 791 678 420 650 364 479 462 592 879 512 105 601 848 31 361 689 444 627 736 866 427 23 138
799 60 81 519 568 290 857 319 272 950 800 303 538 851 127 838 602 286 6 94 194 975 159 169 380
98 771 318 320 462 986 645 578 452 369 823 480 318 911 386 916 130 147 287 524 555 806 667 456
386 179 548 651 19 1000 731 126 328 681 623 216 449 997 495 530 642 917 649 277 213 640 715 472
13 128 776 704 147 281 162 722 591 612 861 329 675 661 897 749 559 145 283 827 716 79 849 404
422 540 594 572 284 401 72 490 430 437 921 889 47 237 524 812 494 454 378 72 823 77 898 492 513
264 888 124 733 327 25 508 749 502 127 988 185 615 447 57 686 885 614 574 583 21 836 485 328
236 76 103 604 470 193 834 674 983 734 846 35 691 527 408 204 393 905 353 370 720 992 906 831
445 238 38 811 117 806 611 453 372 906 905 700 16 263 732 655 876 933 630 252 367 877 709 792
803 278 862 615 323 500 707 477 57 903 738 894 291 350 701 443 530 499 42 863 616 496 95 179
269 311 566 325 981 954 739 218 580 451 565 528 369 532 457 845 673 690 17 748 87 493 86 432
323 84 556 396 910 439 94 421 431 26 936 743 853 990 53 996 52 923 516 253 665 289 791 429 159
571 6 27 36 714 961 308 478 107 928 610 242 723 846 834 696 543 48 488 314 557 741 777 913 77
161 793 716 837 264 389 738 163 883 789 682 620 844 819 754 600 629 474 216 221 740 997 361 349
520 956 96 726 273 765 150 175 272 951 98 232 332 715 645 805 145 309 76 621 249 476 774 273
545 493 53 371 646 541 381 901 750 166 963 726 80 585 912 895 338 71 705 388 409 384 882 629
440 380 96 802 913 704 531 115 544 898 182 143 414 754 73 552 766 891 223 659 73 504 167 261
141 111 188 408 487 657 953 957 612 471 689 605 619 340 816 510 955 643 692 5 306 411 518 423
589 338 245 844 188 652 536 315 551 564 884 593 850 430 566 759 495 984 387 483 595 209 777 313
798 984 874 650 489 196 860 866 370 133 121 482 21 78 130 959 280 878 286 535 775 341 420 195
603 412 191 785 191 39 815 778 340 250 153 828 757 171 91 687 413 677 426 724 755 199 576 465
812 920 838 896 550 455 426 688 201 705 152 324 346 476 928 161 481 902 817 820 941 748 680 213
69 665 79 618 779 802 958 203 212 719 977 424 641 443 383 680 911 788 795 110 150 363 215 131
954 195 840 595 350 894 654 932 184 843 19 67 118 4 25 119 925 804 197 364 745 970 403 854 755
325 391 18 229 735 376 900 176 973 402 915 229 108 684 553 339 672 713 934 546 101 980 48 746
192 111 167 359 224 233 316 65 349 463 137 438 636 868 721 796 259 1 291 521 499 785 29 985 68
635 62 13 960 570 257 773 768 626 335 58 932 706 139 605 263 231 606 205 679 109 134 914 667
988 165 853 358 743 935 320 698 939 631 247 583 828 412 664 888 690 918 212 300 646 966 973 300
317 763 206 209 15 938 835 556 265 32 427 509 593 7 747 781 560 423 896 375 384 120 44 590 414
397 744 559 307 362 204 234 68 893 164 136 682 477 383 144 794 49 465 729 368 305 522 170 979
411 413 113 700 952 20 202 207 27 243 660 919 547 514 967 82 761 466 41 87 830 214 995 201 244
820 875 475 506 915 357 267 886 261 955 703 852 555 12 283 148 922 972 882 942 675 245 808 64
833 977 394 158 551 416 131 154 924 184 869 7 173 723 708 872 757 497 153 321 679 450 912 3 475
549 351 648 825 354 703 498 299 269 698 337 563 765 97 121 751 978 814 658 969 334 382 10 714
486 262 642 33 270 637 83 260 129 189 923 890 944 23 664 30 44 699 146 459 781 160 9 344 173 139
288 295 236 982 50 152 936 330 732 478 531 669 649 502 8 228 267 265 572 490 943 258 235 591 8
278 82 534 106 29 310 569 61 485 981 2 730 758 631 868 332 336 36 487 31 599 656 466 858 88 804

520 410 635 192 974 529 289 102 574 464 783 437 119 937 764 890 917 406 776 433 312 373 101 540 508 382 994 2 352 230 554 156 826 276 786 771 989 693 702 419 811 56 217 457 780 279 37 148 881 510 450 613 729 788 586 422 996 596 801 214 343 348 878 730 588 93 674 617 51 999 930 418 168 599 685 24 782 839 557 976 220 260 825 441 405 945 372 711 247 803 252 575 581 808 588 947 299 4 343 762 991 246 511 354 469 656 655 666 441 333 773 442 717 637 957 847 903 217 787 938 960 92 129 81 608 484 805 751 634 43 735 515 433 219 108 293 258 444 639 985 858 753 11 303 89 158 355 66 155 231 352 275 407 30 122

Test b)

999

194 974 550 698 990 830 619 666 329 577 731 278 738 343 323 596 629 707 128 706 858 81 989 586 342 342 105 874 819 320 507 423 669 322 450 235 679 933 419 435 923 838 791 771 986 493 446 136 202 991 748 993 339 290 486 366 952 203 188 404 647 333 483 932 780 8 98 923 115 381 668 11 670 759 30 563 193 744 91 433 257 101 330 650 521 847 693 107 82 508 845 508 840 121 584 418 348 750 25 863 737 16 963 812 12 751 154 960 86 158 245 722 29 977 440 75 9 781 649 628 534 97 445 645 35 261 294 693 117 831 17 179 287 569 785 356 878 838 152 552 591 635 827 6 881 720 629 685 653 51 913 551 263 663 155 690 805 864 988 506 825 184 810 457 345 206 519 94 81 677 919 436 274 84 86 241 222 643 793 546 84 824 112 374 779 460 862 910 444 770 104 293 104 490 72 322 217 287 280 741 643 562 577 807 503 968 16 805 223 752 779 879 455 627 62 667 846 962 978 324 875 823 538 294 907 54 56 673 648 943 555 40 631 885 637 146 319 96 872 623 71 395 604 415 863 927 359 607 515 361 265 390 570 465 428 472 755 857 944 886 283 695 427 613 238 851 834 363 282 767 776 630 133 712 818 866 682 466 346 159 198 387 730 334 350 541 44 879 511 19 238 493 880 652 677 573 738 870 704 97 666 360 954 717 777 798 714 528 43 244 350 596 38 409 719 15 455 546 31 946 330 189 934 784 715 29 175 480 234 150 733 818 34 424 326 72 841 633 706 222 978 305 387 145 705 566 527 365 631 503 21 382 349 588 898 328 871 553 155 567 884 505 178 399 917 360 296 466 862 139 187 981 648 43 792 211 436 594 421 295 10 949 520 749 420 881 954 689 883 684 14 889 715 636 800 699 470 435 709 454 354 790 64 731 540 942 579 623 734 209 213 543 136 398 856 286 254 15 929 625 106 855 589 581 7 973 948 361 622 316 453 318 650 291 365 820 936 842 165 76 809 289 463 736 761 518 106 725 980 775 774 953 755 904 743 467 218 857 891 299 137 970 819 617 691 854 875 848 307 398 108 489 482 256 167 514 947 603 272 839 946 253 638 959 477 580 611 619 741 642 124 101 415 57 478 968 37 462 126 899 964 635 953 402 616 153 298 379 135 930 534 769 229 740 905 902 259 70 23 53 180 187 494 58 609 971 542 762 828 228 149 489 858 660 285 264 576 880 703 301 868 344 624 170 549 890 492 928 311 61 515 142 822 141 52 286 728 426 823 713 884 495 329 604 343 323 432 813 782 48 471 900 59 497 618 80 103 122 437 380 408 241 54 633 292 972 372 121 61 102 114 541 893 776 205 718 918 758 992 870 381 190 488 411 268 306 914 783 413 152 574 252 191 92 803 38 511 849 353 312 87 704 752 65 392 481 410 799 3 542 963 672 159 273 290 985 175 942 760 527 480 817 425 647 753 138 701 338 597 434 347 516 697 794 911 914 299 641 364 940 94 259 821 328 36 405 399 636 469 269 271 394 41 459 465 888 164 325 620 614 908 125 349 824 153 727 609 922 816 88 358 589 544 533 851 916 601 501 18 499 31 65 44 975 375 131 981 389 760 347 966 295 913 869 572 772 124 195 173 260 76 566 562 536 670 612 700 939 657 761 377 332 965 117 184 214 102 202 42 379 834 239 683 561 611 678 60 200 154 219 975 265 181 284 385 89 813 627 225 371 393 63 672 518 85 77 830 671 400 906 718 592 424 149 100 756 64 180 687 808 944 787 307 826 901 829 315 729 993 661 820 471 199 759 37 247 720 645 39 876 427 843 32 396 90 45 386 105 523 3 474 355 610 593 189 513 686 204 378 428 876 649 841 859 233 142 103 909 959 156 447 30 565 535 245 728 169 144 248 710 691 778 422 67 811 183 266 324 496 363 853 374 567 517 846 68 565 561 867 585 598 921 352 625 313 39 522 224 167 449 897 134 739 491 723 804 692 89

938 227 593 989 998 837 362 632 899 115 25 513 321 437 668 270 606 235 446 815 157 797 753 257
228 766 351 634 957 526 836 247 582 316 251 406 724 50 95 302 895 430 697 549 940 898 892 694
174 894 309 919 537 507 111 757 607 921 442 34 869 441 573 777 843 951 172 113 889 447 63 397
742 264 654 138 367 10 626 578 945 243 545 216 860 910 733 764 441 443 49 224 327 448 220 601
560 903 895 789 906 853 473 366 789 788 311 749 69 390 472 995 431 336 304 169 36 33 254 632 62
279 231 664 386 642 522 998 198 22 892 763 498 421 8 275 161 371 457 686 735 176 924 182 559
310 579 835 983 888 206 605 950 331 530 999 956 890 9 618 646 218 260 955 748 283 300 826 412
839 442 11 667 410 317 401 833 502 129 230 416 368 624 377 659 726 652 558 802 80 258 615 368
417 550 384 248 145 109 937 544 487 293 509 808 556 949 560 160 20 233 690 680 939 972 277 487
407 364 510 815 246 357 897 383 590 638 229 747 872 804 27 866 444 263 320 786 26 400 646 458
370 564 492 931 925 768 185 190 310 332 403 440 73 150 140 69 300 100 482 896 331 250 976 132
434 539 250 135 78 557 514 403 346 422 904 903 770 797 74 75 491 369 524 111 201 351 193 413
874 296 969 796 341 281 113 333 750 464 676 404 960 116 339 110 378 709 859 262 479 420 568 526
886 926 861 77 694 928 816 771 458 995 230 873 932 956 4 252 610 359 476 338 83 854 122 128 674
353 568 289 817 315 160 161 901 774 525 850 429 46 83 580 984 702 1 120 688 671 383 982 827 745
615 302 125 612 385 597 821 669 2 485 479 828 784 68 912 710 207 303 242 314 742 556 512 911
337 120 475 298 232 662 88 500 739 586 569 581 833 814 716 848 997 127 251 598 639 28 531 93
675 936 199 571 585 997 114 850 380 722 740 509 240 844 958 459 18 721 92 966 571 769 832 588
765 639 758 23 766 370 215 707 474 524 900 861 606 822 55 335 641 17 934 717 922 743 734 617
673 982 312 91 118 683 394 317 42 438 143 212 174 261 948 684 358 891 318 554 950 719 852 533
682 273 983 388 367 965 658 662 678 699 894 536 429 500 96 357 95 600 288 21 938 469 915 28 996
974 935 726 701 916 696 679 177 637 721 987 53 449 695 478 798 840 119 33 123 301 681 452 991
408 737 412 873 781 200 216 961 941 431 12 675 832 994 56 751 46 992 71 48 979 855 170 267 665
344 723 712 548 384 495 127 116 497 443 622 297 57 319 994 275 806 147 314 255 498 925 141 595
5 411 772 234 306 730 223 801 856 4 521 785 79 242 700 243 762 27 945 553 5 192 183 620 405 878
716 689 418 140 754 517 674 745 197 55 905 484 246 773 87 578 977 205 292 660 773 987 871 653
195 13 473 255 969 391 947 962 236 173 490 902 340 535 563 450 22 552 941 548 51 496 282 893
157 547 621 208 212 166 525 90 486 215 867 538 477 714 470 213 236 271 736 132 558 655 576 802
510 129 279 402 276 151 375 163 793 520 836 191 523 207 803 528 99 795 917 935 545 1 451 58 144
433 659 308 661 119 757 203 908 362 389 599 334 608 219 168 196 952 929 937 476 214 680 551 345
182 787 414 844 24 504 438 227 831 814 267 531 554 461 484 276 575 453 512 204 220 304 591 7
602 688 396 225 178 463 126 988 658 795 158 756 835 976 931 530 613 868 237 397 887 221 711 847
2 448 177 537 685 924 809 663 6 59 485 783 595 746 961 796 529 640 269 20 373 951 93 628 45 321
369 78 249 788 735 454 920 696 423 249 651 419 146 702 918 70 131 172 221 602 594 309 582 274
171 285 237 288 599 40 801 417 516 852 865 231 732 416 355 614 284 99 240 782 727 587 705 186
197 468 414 210 134 26 764 445 348 426 196 137 575 572 600 462 608 392 713 47 226 605 256 780
812 372 933 270 336 687 109 407 143 382 258 676 767 807 920 532 765 208 494 73 464 148 882 209
337 501 41 547 806 664 794 681 452 729 499 123 47 49 885 555 19 66 778 168 708 24 108 799 909
162 461 409 13 703 985 376 529 98 217 272 725 979 468 432 996 967 430 590 519 326 711 303 85 52
297 171 837 877 630 754 829 391 986 877 281 327 112 107 626 810 543 915 654 860 239 583 253 584
973 325 656 130 746 456 60 644 481 277 201 483 176 592 763 786 849 356 291 943 621 927 14 574
280 133 439 505 162 811 32 865 118 502 665 864 926 504 50 166 393 955 634 165 79 603 475 970
388 912 557 74 308 570 262 842 984 110 186 698 616 971 460 644 999 139 185 335 456 747 958 506
540 907 964 967 376 194 232 66 341 724 768 164 406 640 181 35 957 708 354 559 163 744 732 82
845 188 467 425 211 790 130 539 352 657 791 882 340 692 156 587 656 268 226 980 655 151 266 990
373 564 451 792 67 278 887 896 305 395 179 488 583 313 930 192 651 800 401 825 775 883 244 439
532 210 148 147