7. praktiskais darbs. 2. semestris

1. uzdevums

Lietotājs ievada divus dažāda garuma augošā secībā sakārtotus masīvus. Lineārā laikā apvienot abus masīvus vienā augoša secībā sakārtotā masīvā un izvadīt to uz ekrāna.

Kods:

```
# Programmas nosaukums: Apvienot divus masīvus augoša secībā lineārā laikā

# 1. uzdevums (1MPR07_Vladislavs_Babaņins)

# Uzdevuma formulējums: Lietotājs ievada divus dažāda garuma augošā secībā sakārtotus masīvus.

# Lineārā laikā apvienot abus masīvus vienā augoša secībā sakārtotā masīvā un izvadīt to uz ekrāna.

# Versija 1.0


import numpy



def izveidot_masivu_ar_garumu(n):
    # Izveido masīvu ar noradīto garumu n
    # n - naturāls skaitlis
    a = numpy.arange(n)
    for i in range(n):
        b = input("Ievadiet " + str(i) + ".elementu ===> ")
        b = is_whole(b, i)
        a[i] = b
    return a



def izvade(x):
    # Izvada masīva elementus pēc kārtas līdz pedējam
```

```
    # x - viendimensijas masīvs
    n = len(x)
    s = str(x[0])
    for i in range(1, n):
        s = s + ", " + str(x[i])
    print(s)


def is_whole(x, i):
    # Pārbauda vai simbolu virkne ir vesels skaitlis un ja nē, tad paprasa ievadīt to vēlreiz (bezgalīgi daudz ievades)
    # Ja simbolu virkne ir vesels skaitlis, tad atgriež to kā int(x)
    # x - pārbaudāma simbolu virkne
    # i - i-tajs elements
    while True:
        try:
            x = int(x)
        except:
            x = input("Kļūda! Ievadiet " + str(i) + ".elementu ===> ")
        else:
            return int(x)


def is_natural(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav
    # Ja ir naturāls skaitlis, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    if str(n).isdigit() and float(n) == int(n) and int(n) > 0:
        return True
    else:
        return False
```

```
def apvieno(a, b):
    # Apvieno divus sakartotus masīvus a un b, un sakārto tos
    # Lineāra laiks o(n)
    # a - viendimensijas masīvs
    # b - viendimensijas masīvs
    ga = len(a)
    gb = len(b)
    gc = ga + gb
    c = numpy.arange(gc)
    ia = 0
    ib = 0
    ic = 0
    while (ia < ga) and (ib < gb):
        if a[ia] < b[ib]:
            c[ic] = a[ia]
            ia = ia + 1
        else:
            c[ic] = b[ib]
            ib = ib + 1
        ic = ic + 1
    if ia < ga:
        for i in range(ia, ga):
            c[ic] = a[i]
            ic = ic + 1
    else:
        for i in range(ib, gb):
            c[ic] = b[i]
            ic = ic + 1
    return c
```

```
# ------------------------------------------------------
# Galvenā programmas daļa
# ------------------------------------------------------


m = input("Ievadiet 1. masīva izmēru ===> ")

while is_natural(m) == False:
    m = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

m = int(m)

print("\nIevadiet sakārota augošā secība masīva skaitļus!")
a = izveidot_masivu_ar_garumu(m)


n = input("\nIevadiet 2. masīva izmēru ===> ")

while is_natural(n) == False:
    n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

n = int(n)

print("\nIevadiet sakārota augošā secība masīva skaitļus!")
b = izveidot_masivu_ar_garumu(n)

print("\nPirmais augošā secība sakārtots masīvs:")
izvade(a)
```

```
print("\nOtrais augošā secība sakārtots masīvs:")
izvade(b)

c = apvieno(a, b)

print("\nApvienots augošā secība sakārtots masīvs:")
izvade(c)
```

Testa piemēri:

1)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Ievadiet 2. masīva izmēru ===> 3

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Pirmais augošā secība sakārtots masīvs:
1, 2, 3

Otrais augošā secība sakārtots masīvs:
1, 2, 3

Apvienots augošā secība sakārtots masīvs:
1, 1, 2, 2, 3, 3
```

2)

```
Ievadiet 1. masīva izmēru ===> 2

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 6
Ievadiet 1.elementu ===> 8

Ievadiet 2. masīva izmēru ===> 4

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> -1
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> 2
Ievadiet 3.elementu ===> 4

Pirmais augošā secība sakārtots masīvs:
6, 8

Otrais augošā secība sakārtots masīvs:
-1, 0, 2, 4

Apvienots augošā secība sakārtots masīvs:
-1, 0, 2, 4, 6, 8
```

3)

```
Ievadiet 1. masīva izmēru ===> 0
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> -1
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 12.5
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> pieci
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 5

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3
Ievadiet 3.elementu ===> 4
Ievadiet 4.elementu ===> 5

Ievadiet 2. masīva izmēru ===> 3

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 12.4
Kļūda! Ievadiet 0.elementu ===> 3
Ievadiet 1.elementu ===> 4
Ievadiet 2.elementu ===> 5

Pirmais augošā secība sakārtots masīvs:
1, 2, 3, 4, 5

Otrais augošā secība sakārtots masīvs:
3, 4, 5

Apvienots augošā secība sakārtots masīvs:
1, 2, 3, 3, 4, 4, 5, 5
```

4)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 0
Ievadiet 1.elementu ===> 5
Ievadiet 2.elementu ===> 10

Ievadiet 2. masīva izmēru ===> 5

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 0
Ievadiet 1.elementu ===> 1
Ievadiet 2.elementu ===> 6
Ievadiet 3.elementu ===> 7
Ievadiet 4.elementu ===> 8

Pirmais augošā secība sakārtots masīvs:
0, 5, 10

Otrais augošā secība sakārtots masīvs:
0, 1, 6, 7, 8

Apvienots augošā secība sakārtots masīvs:
0, 0, 1, 5, 6, 7, 8, 10
```

5)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> 0
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> 0

Ievadiet 2. masīva izmēru ===> 5

Ievadiet sakārota augošā secība masīva skaitļus!
Ievadiet 0.elementu ===> -1
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> 1
Ievadiet 3.elementu ===> 2
Ievadiet 4.elementu ===> 66

Pirmais augošā secība sakārtots masīvs:
0, 0, 0

Otrais augošā secība sakārtots masīvs:
-1, 0, 1, 2, 66

Apvienots augošā secība sakārtots masīvs:
-1, 0, 0, 0, 0, 1, 2, 66
```

2. uzdevums

Lietotājs ievada trīs dažāda garuma masīvus, no kuriem pirmais ir sakārtots augošā secībā, otrais ir sakārtots dilstošā secībā un trešais sakārtots augošā secībā. Lineārā laikā apvienot visus trīs masīvus vienā dilstošā secībā sakārtotā masīvā un izvadīt to uz ekrāna.

Kods:

```
# Programmas nosaukums: Trīs masīvu apvienošana. 1.-augošs, 2.-dilstošs, 3.-augošs

# 2. uzdevums (1MPR07_Vladislavs_Babaņins)

# Uzdevuma formulējums: Lietotājs ievada trīs dažāda garuma masīvus, no kuriem pirmais ir sakārtots augošā secībā,

# otrais ir sakārtots dilstošā secībā un trešais sakārtots augošā secībā.

# Lineārā laikā apvienot visus trīs masīvus vienā dilstošā secībā sakārtotā masīvā un izvadīt to uz ekrāna.

# Versija 1.0



import numpy



def izveidot_masivu_ar_garumu(n):
    # Izveido masīvu ar noradīto garumu n
    # n - naturāls skaitlis
    a = numpy.arange(n)
    for i in range(n):
        b = input("Ievadiet " + str(i) + ".elementu ===> ")
        b = is_whole(b, i)
        a[i] = b
    return a



def izvade(x):
    # Izvada masīva elementus pēc kārtas līdz pedējam
    # x - viendimensijas masīvs
```

```
    n = len(x)
    s = str(x[0])
    for i in range(1, n):
        s = s + ", " + str(x[i])
    print(s)


def is_whole(x, i):
    # Pārbauda vai simbolu virkne ir vesels skaitlis un ja nē, tad paprasa ievadīt to vēlreiz (bezgalīgi daudz ievādes)
    # Ja simbolu virkne ir vesels skaitlis, tad atgriež to kā int(x)
    # x - pārbaudāma simbolu virkne
    # i - i-tajs elements
    while True:
        try:
            x = int(x)
        except:
            x = input("Kļūda! Ievadiet " + str(i) + ".elementu ===> ")
        else:
            return int(x)


def is_natural(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav
    # Ja ir naturāls skaitlis, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    if str(n).isdigit() and float(n) == int(n) and int(n) > 0:
        return True
    else:
        return False
```

```
def apvieno(a, b):
    # Apvieno divus sakartotus masīvus a un b, un sakarto tos
    # Lineāra laiks o(n)
    # a - viendimensijas masīvs
    # b - viendimensijas masīvs
    ga = len(a)
    gb = len(b)
    gc = ga + gb
    c = numpy.arange(gc)
    ia = 0
    ib = 0
    ic = 0
    while (ia < ga) and (ib < gb):
        if a[ia] < b[ib]:
            c[ic] = a[ia]
            ia = ia + 1
        else:
            c[ic] = b[ib]
            ib = ib + 1
        ic = ic + 1
    if ia < ga:
        for i in range(ia, ga):
            c[ic] = a[i]
            ic = ic + 1
    else:
        for i in range(ib, gb):
            c[ic] = b[i]
            ic = ic + 1
    return c
```

```
def reverse(masivs):
    # Pārkarto masīvā visus elementus pretēji
    # masivs - viendimensijas masīvs
    start_index = 0
    end_index = len(masivs) - 1
    while end_index > start_index:
        temp = masivs[start_index]
        masivs[start_index] = masivs[end_index]
        masivs[end_index] = temp
        start_index = start_index + 1
        end_index = end_index - 1



# ------------------------------------------------------------
# Galvenā programmas daļa
# ------------------------------------------------------------



m = input("Ievadiet 1. masīva izmēru ===> ")

while is_natural(m) == False:
    m = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

m = int(m)

print("\nIevadiet sakārota augošā (nedilstoša) secība masīva skaitļus!")
a = izveidot_masivu_ar_garumu(m)



n = input("\nIevadiet 2. masīva izmēru ===> ")
```

```
while is_natural(n) == False:
    n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

n = int(n)

print("\nIevadiet sakārota dilstoša (neaugoša) secība masīva skaitļus!")
b = izveidot_masivu_ar_garumu(n)



k = input("\nIevadiet 3. masīva izmēru ===> ")

while is_natural(k) == False:
    n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

k = int(k)

print("\nIevadiet sakārota augošā (nedilstoša) secība masīva skaitļus!")
c = izveidot_masivu_ar_garumu(k)



print("\nPirmais augošā (nedilstoša) secība sakārtots masīvs:")
izvade(a)
print("\nOtrais dilstoša (neaugoša) secība sakārtots masīvs:")
izvade(b)
print("\nTrešais augošā (nedilstoša) secība sakārtots masīvs:")
izvade(c)



reverse(b)
```

```
ab = apvieno(a, b)

abc = apvieno(ab, c)


reverse(abc)

print("\nApvienoti visi trīs sakārtoti masīvi dilstoša (neaugoša) secība:")

izvade(abc)
```

Testa piemēri:

1)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Ievadiet 2. masīva izmēru ===> 4

Ievadiet sakārota dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 6
Ievadiet 1.elementu ===> 5
Ievadiet 2.elementu ===> 4
Ievadiet 3.elementu ===> 3

Ievadiet 3. masīva izmēru ===> 2

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 2
Ievadiet 1.elementu ===> 20

Pirmais augošā (nedilstoša) secība sakārtots masīvs:
1, 2, 3

Otrais dilstoša (neaugoša) secība sakārtots masīvs:
6, 5, 4, 3

Trešais augošā (nedilstoša) secība sakārtots masīvs:
2, 20

Apvienoti visi trīs sakārtoti masīvi dilstoša (neaugoša) secība:
20, 6, 5, 4, 3, 3, 2, 2, 1
```

2)

```
Ievadiet 1. masīva izmēru ===> 5

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> -3
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> 3
Ievadiet 3.elementu ===> 6
Ievadiet 4.elementu ===> 8

Ievadiet 2. masīva izmēru ===> 0
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> -1
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 12.4
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> pieci
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 5

Ievadiet sakārota dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> -1
Ievadiet 1.elementu ===> pieci
Kļūda! Ievadiet 1.elementu ===> 1
Ievadiet 2.elementu ===> 1
Ievadiet 3.elementu ===> 1
Ievadiet 4.elementu ===> 1

Ievadiet 3. masīva izmēru ===> 1

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 5000000

Pirmais augošā (nedilstoša) secība sakārtots masīvs:
-3, 0, 3, 6, 8

Otrais dilstoša (neaugoša) secība sakārtots masīvs:
-1, 1, 1, 1, 1

Trešais augošā (nedilstoša) secība sakārtots masīvs:
5000000

Apvienoti visi trīs sakārtoti masīvi dilstoša (neaugoša) secība:
5000000, 8, 6, 3, -1, 1, 1, 1, 1, 0, -3
```

3)

```
Ievadiet 1. masīva izmēru ===> 1

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> -4

Ievadiet 2. masīva izmēru ===> 1

Ievadiet sakārota dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> -100

Ievadiet 3. masīva izmēru ===> 1

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 100

Pirmais augošā (nedilstoša) secība sakārtots masīvs:
-4

Otrais dilstoša (neaugoša) secība sakārtots masīvs:
-100

Trešais augošā (nedilstoša) secība sakārtots masīvs:
100

Apvienoti visi trīs sakārtoti masīvi dilstoša (neaugoša) secība:
100, -4, -100
```

4)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Ievadiet 2. masīva izmēru ===> 4

Ievadiet sakārota dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 0
Ievadiet 1.elementu ===> -3
Ievadiet 2.elementu ===> -7
Ievadiet 3.elementu ===> -10

Ievadiet 3. masīva izmēru ===> 2

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 50000000

Pirmais augošā (nedilstoša) secība sakārtots masīvs:
1, 2, 3

Otrais dilstoša (neaugoša) secība sakārtots masīvs:
0, -3, -7, -10

Trešais augošā (nedilstoša) secība sakārtots masīvs:
1, 50000000

Apvienoti visi trīs sakārtoti masīvi dilstoša (neaugoša) secība:
50000000, 3, 2, 1, 1, 0, -3, -7, -10
```

5)

```
Ievadiet 1. masīva izmēru ===> 4

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 0
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> 0
Ievadiet 3.elementu ===> 0

Ievadiet 2. masīva izmēru ===> 4

Ievadiet sakārota dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> -2
Ievadiet 3.elementu ===> -4

Ievadiet 3. masīva izmēru ===> 1

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> -12312313

Pirmais augošā (nedilstoša) secība sakārtots masīvs:
0, 0, 0, 0

Otrais dilstoša (neaugoša) secība sakārtots masīvs:
1, 0, -2, -4

Trešais augošā (nedilstoša) secība sakārtots masīvs:
-12312313

Apvienoti visi trīs sakārtoti masīvi dilstoša (neaugoša) secība:
1, 0, 0, 0, 0, 0, -2, -4, -12312313
```

6)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Ievadiet 2. masīva izmēru ===> 3

Ievadiet sakārota dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 3
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 1

Ievadiet 3. masīva izmēru ===> 3

Ievadiet sakārota augošā (nedilstoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Pirmais augošā (nedilstoša) secība sakārtots masīvs:
1, 2, 3

Otrais dilstoša (neaugoša) secība sakārtots masīvs:
3, 2, 1

Trešais augošā (nedilstoša) secība sakārtots masīvs:
1, 2, 3

Apvienoti visi trīs sakārtoti masīvi dilstoša (neaugoša) secība:
3, 3, 3, 2, 2, 2, 1, 1, 1
```

PU1. uzdevums

Lietotājs ievada trīs sakārtotus masīvus, bet nav zināms, kurš no tiem ir sakārtots augošā vai dilstošā secībā. Lineārā laikā apvienot visus trīs masīvus vienā augošā secībā sakārtotā masīvā un izvadīt to uz ekrāna.

Kods:

```
# Programmas nosaukums: Trīs masīvu apvienošana

# Papilduzdevums 1 (1MPR07_Vladislavs_Babaņins)

# Uzdevuma formulējums: Lietotājs ievada trīs sakārtotus masīvus, bet nav zināms, kurš no tiem ir sakārtots augošā vai dilstošā secībā.

# Lineārā laikā apvienot visus trīs masīvus vienā augošā secībā sakārtotā masīvā un izvadīt to uz ekrāna.

# Versija 1.0



import numpy



def izveidot_masivu_ar_garumu(n):
    # Izveido masīvu ar noradīto garumu n
    # n - naturāls skaitlis
    a = numpy.arange(n)
    for i in range(n):
        b = input("Ievadiet " + str(i) + ".elementu ===> ")
        b = is_whole(b, i)
        a[i] = b
    return a



def izvade(x):
    # Izvada masīva elementus pēc kārtas līdz pēdējam
    # x - viendimensijas masīvs
    n = len(x)
```

```
    s = str(x[0])
    for i in range(1, n):
        s = s + ", " + str(x[i])
    print(s)


def is_whole(x, i):
    # Pārbauda vai simbolu virkne ir vesels skaitlis un ja nē, tad paprasa ievadīt to vēlreiz (bezgalīgi daudz ievades)
    # Ja simbolu virkne ir vesels skaitlis, tad atgriež to kā int(x)
    # x - pārbaudāma simbolu virkne
    # i - i-tajs elements
    while True:
        try:
            x = int(x)
        except:
            x = input("Kļūda! Ievadiet " + str(i) + ".elementu ===> ")
        else:
            return int(x)


def is_natural(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav
    # Ja ir naturāls skaitlis, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    if str(n).isdigit() and float(n) == int(n) and int(n) > 0:
        return True
    else:
        return False
```

```
def apvieno(a, b):
    # Apvieno divus sakārtotus masīvus a un b, un sakārto tos
    # Ātrums - lineārs laiks o(n)
    # a - viendimensijas masīvs
    # b - viendimensijas masīvs
    ga = len(a)
    gb = len(b)
    gc = ga + gb
    c = numpy.arange(gc)
    ia = 0
    ib = 0
    ic = 0
    while (ia < ga) and (ib < gb):
        if a[ia] < b[ib]:
            c[ic] = a[ia]
            ia = ia + 1
        else:
            c[ic] = b[ib]
            ib = ib + 1
        ic = ic + 1
    if ia < ga:
        for i in range(ia, ga):
            c[ic] = a[i]
            ic = ic + 1
    else:
        for i in range(ib, gb):
            c[ic] = b[i]
            ic = ic + 1
    return c
```

```
def reverse(masivs):
    # Pārkarto masīvā visus elementus pretēji
    # masivs - viendimensijas masīvs
    start_index = 0
    end_index = len(masivs) - 1

    while end_index > start_index:
        temp = masivs[start_index]
        masivs[start_index] = masivs[end_index]
        masivs[end_index] = temp

        start_index = start_index + 1
        end_index = end_index - 1

    return masivs



def is_ascending(n):  # vai nedilstoša?
    # Pārbauda vai masīvs ir augošs (nedilstošs)
    # Ja masīvs ir nedilstošs (nav augošs), tad return True
    # Ja masīvs nav nedilstošs (nav augošs), tad return False
    # n - viendimensijas masīvs
    if len(n) == 1:
        return True

    for i in range(0, len(n)):
        if i < len(n) - 1 and n[i] > n[i + 1]:
            return False  # Nē, nav augoša, nav nedilstoša, nav konstanta (varētu būt augoša, vai
nekāda)

    return True  # Jā, ir augoša (vai nedilstoša)
```

```
def is_descending(n):  # vai neaugoša?
    # Pārbauda vai masīvs ir dilstošs (neaugošs)
    # Ja masīvs ir neaugošs (nav dilstošs), tad return True
    # Ja masīvs nav neaugošs (nav dilstošs), tad return False
    # n - viendimensijas masīvs
    if len(n) == 1:
        return True

    for i in range(0, len(n)):
        if i < len(n) - 1 and n[i] < n[i + 1]:
            return False  # Nē, nav dilstoša, nav neaugoša, nav konstanta (varētu būt augoša, vai nekāda)

    return True  # Jā, ir dilstoša (vai neaugoša)


# ------------------------------------------------------
# Galvenā programmas daļa
# ------------------------------------------------------


m = input("Ievadiet 1. masīva izmēru ===> ")

while is_natural(m) == False:
    m = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

m = int(m)

print("\nIevadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!")
```

```
a = izveidot_masivu_ar_garumu(m)

if is_ascending(a) == False and is_descending(a) == False:  # tad nekāda
    print("\nKļūda!\nPirmais ievadītais masīvs:")
    izvade(a)
    print("Nav ne augoši sakārtots, nav ne dilstoši sakārtots!")
    quit()



n = input("\nIevadiet 2. masīva izmēru ===> ")

while is_natural(n) == False:
    n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

n = int(n)

print("Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!")
b = izveidot_masivu_ar_garumu(n)

if is_ascending(b) == False and is_descending(b) == False:
    print("\nKļūda!\nOtrais ievadītais masīvs:")
    izvade(b)
    print("Nav ne augoši sakārtots, nav ne dilstoši sakārtots!")
    quit()



k = input("\nIevadiet 3. masīva izmēru ===> ")

while is_natural(k) == False:
    n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")
```

```
k = int(k)

print("\nIevadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!")
c = izveidot_masivu_ar_garumu(k)

if is_ascending(c) == False and is_descending(c) == False:
    print("\nKļūda!\nTrešais ievadītais masīvs:")
    izvade(c)
    print("Nav ne augoši sakārtots, nav ne dilstoši sakārtots!")
    quit()



print("\nPirmais sakārtots masīvs:")
izvade(a)
print("\nOtrais sakārtots masīvs:")
izvade(b)
print("\nTrešais sakārtots masīvs:")
izvade(c)



if is_ascending(a) == False:
    reverse(a)
if is_ascending(b) == False:
    reverse(b)
if is_ascending(c) == False:
    reverse(c)



ab = apvieno(a, b)
abc = apvieno(ab, c)
```

print("\nApvienots augošā (nedilstoša) secība sakārtots masīvs:")

izvade(abc)

Testa piemēri:

1)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Ievadiet 2. masīva izmēru ===> 1
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 3

Ievadiet 3. masīva izmēru ===> 3

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Pirmais sakārtots masīvs:
1, 2, 3

Otrais sakārtots masīvs:
3

Trešais sakārtots masīvs:
1, 2, 3

Apvienots augošā (nedilstoša) secība sakārtots masīvs:
1, 1, 2, 2, 3, 3, 3
```

2)

```
Ievadiet 1. masīva izmēru ===> -5
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 0
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> pieci
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 5

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3
Ievadiet 3.elementu ===> 4
Ievadiet 4.elementu ===> 5

Ievadiet 2. masīva izmēru ===> 2
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 10
Ievadiet 1.elementu ===> 1

Ievadiet 3. masīva izmēru ===> 3

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> -1
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> 1

Pirmais sakārtots masīvs:
1, 2, 3, 4, 5

Otrais sakārtots masīvs:
10, 1

Trešais sakārtots masīvs:
-1, 0, 1

Apvienots augošā (nedilstoša) secība sakārtots masīvs:
-1, 0, 1, 1, 1, 2, 3, 4, 5, 10
```

3)

```
Ievadiet 1. masīva izmēru ===> 1

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1

Ievadiet 2. masīva izmēru ===> 1
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 2

Ievadiet 3. masīva izmēru ===> 1

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> -20

Pirmais sakārtots masīvs:
1

Otrais sakārtots masīvs:
2

Trešais sakārtots masīvs:
-20

Apvienots augošā (nedilstoša) secība sakārtots masīvs:
-20, 1, 2
```

4)

```
Ievadiet 1. masīva izmēru ===> 4

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3
Ievadiet 3.elementu ===> 4

Ievadiet 2. masīva izmēru ===> 4
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 6
Ievadiet 1.elementu ===> 5
Ievadiet 2.elementu ===> 4
Ievadiet 3.elementu ===> 3

Ievadiet 3. masīva izmēru ===> 2

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 10000
Ievadiet 1.elementu ===> 222

Pirmais sakārtots masīvs:
1, 2, 3, 4

Otrais sakārtots masīvs:
6, 5, 4, 3

Trešais sakārtots masīvs:
10000, 222

Apvienots augošā (nedilstoša) secība sakārtots masīvs:
1, 2, 3, 3, 4, 4, 5, 6, 222, 10000
```

5)

```
Ievadiet 1. masīva izmēru ===> 2

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 12
Ievadiet 1.elementu ===> 13

Ievadiet 2. masīva izmēru ===> 5
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 5
Ievadiet 2.elementu ===> 6
Ievadiet 3.elementu ===> 1000
Ievadiet 4.elementu ===> 1000000

Ievadiet 3. masīva izmēru ===> 3

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 0
Ievadiet 1.elementu ===> -1
Ievadiet 2.elementu ===> -10

Pirmais sakārtots masīvs:
12, 13

Otrais sakārtots masīvs:
1, 5, 6, 1000, 1000000

Trešais sakārtots masīvs:
0, -1, -10

Apvienots augošā (nedilstoša) secība sakārtots masīvs:
-10, -1, 0, 1, 5, 6, 12, 13, 1000, 1000000
```

6)

```
Ievadiet 1. masīva izmēru ===> 3

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 3
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 1

Ievadiet 2. masīva izmēru ===> 3
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 2
Ievadiet 1.elementu ===> 1
Ievadiet 2.elementu ===> 3

Kļūda!
Otrais ievadītais masīvs:
2, 1, 3
Nav ne augoši sakārtots, nav ne dilstoši sakārtots!
```

7)

```
Ievadiet 1. masīva izmēru ===> 6

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 5
Ievadiet 2.elementu ===> 2
Ievadiet 3.elementu ===> 3
Ievadiet 4.elementu ===> 6
Ievadiet 5.elementu ===> 8

Kļūda!
Pirmais ievadītais masīvs:
1, 5, 2, 3, 6, 8
Nav ne augoši sakārtots, nav ne dilstoši sakārtots!
```

8)

```
Ievadiet 1. masīva izmēru ===> 4

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 2
Ievadiet 1.elementu ===> 3
Ievadiet 2.elementu ===> 6
Ievadiet 3.elementu ===> 8

Ievadiet 2. masīva izmēru ===> 2
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 2
Ievadiet 1.elementu ===> 2

Ievadiet 3. masīva izmēru ===> 4

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 6
Ievadiet 1.elementu ===> 5
Ievadiet 2.elementu ===> 4
Ievadiet 3.elementu ===> 3

Pirmais sakārtots masīvs:
2, 3, 6, 8

Otrais sakārtots masīvs:
2, 2

Trešais sakārtots masīvs:
6, 5, 4, 3

Apvienots augošā (nedilstoša) secība sakārtots masīvs:
2, 2, 2, 3, 3, 4, 5, 6, 6, 8
```

9)

```
Ievadiet 1. masīva izmēru ===> 4

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 2
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 2
Ievadiet 3.elementu ===> 2

Ievadiet 2. masīva izmēru ===> 3
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 3
Ievadiet 2.elementu ===> 5

Ievadiet 3. masīva izmēru ===> 2

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 6

Pirmais sakārtots masīvs:
2, 2, 2, 2

Otrais sakārtots masīvs:
1, 3, 5

Trešais sakārtots masīvs:
1, 6

Apvienots augošā (nedilstoša) secība sakārtots masīvs:
1, 1, 2, 2, 2, 2, 3, 5, 6
```

10)

```
Ievadiet 1. masīva izmēru ===> 0
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> pieci
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> -2
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 2

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 12.2
Kļūda! Ievadiet 0.elementu ===> pieci
Kļūda! Ievadiet 0.elementu ===> 5
Ievadiet 1.elementu ===> 6

Ievadiet 2. masīva izmēru ===> 3
Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 0
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> 3

Ievadiet 3. masīva izmēru ===> 3

Ievadiet sakārota augoša (nedilstoša) vai dilstoša (neaugoša) secība masīva skaitļus!
Ievadiet 0.elementu ===> 3
Ievadiet 1.elementu ===> 7
Ievadiet 2.elementu ===> 1

Kļūda!
Trešais ievadītais masīvs:
3, 7, 1
Nav ne augoši sakārtots, nav ne dilstoši sakārtots!
```

3. uzdevums

Sastādīt programmu, kas realizē saliešanas (izmantojot skaldi un valdi) kārtošanas metodi.

Norādījumi uzdevuma risinājuma izveidei:

- Nedrīkst rakstīt algoritmu, kas vispirms sakārto masīvu augošā secībā un pēc tam to nodrukā apgrieztā secībā!
- Katrā kārtošanas algoritma kods jāpapildina tā, lai tiktu saskaitīts masīva elementu salīdzināšanas reižu skaits.
- Lai salīdzinātu salīdzināšanas reižu skaitu katrai metodei, ar katru no metodēm jākārto viens un tas pats masīvs.
- Salīdzināt iegūtos rezultātus ar iepriekšējos praktiskajos darbos apskatītajiem kārtošanas algoritmiem.
- Programmas izpildes rezultātā uz ekrāna tiek parādīti abu kārtošanas metožu izpildes rezultāti, sakārtojot 10, 1 000, 10 000 skaitļus.

Kods:

```
# Programmas nosaukums: Saliešanas

# 3. uzdevums (1MPR07_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē saliešanas (izmantojot "Skaldi un valdi") kārtošanas metodi.

# Norādījumi uzdevuma risinājuma izveidei:

# Nedrīkst rakstīt algoritmu, kas vispirms sakārto masīvu augošā secībā un pēc tam to nodrukā apgrieztā secībā!

# Katrā kārtošanas algoritma kods jāpapildina tā, lai tiktu saskaitīts masīva elementu salīdzināšanas reižu skaits.

# Lai salīdzinātu salīdzināšanas reižu skaitu katrai metodei, ar katru no metodēm jākārto viens un tas pats masīvs.

# Salīdzināt iegūtos rezultātus ar iepriekšējos praktiskajos darbos apskatītajiem kārtošanas algoritmiem.

# Programmas izpildes rezultātā uz ekrāna tiek parādīti abu kārtošanas metožu izpildes rezultāti, sakārtojot 10, 1 000, 10 000 skaitļus.

# Versija 1.0



import numpy

import random

import math
```

```
import copy



def skaldi_un_valdi_dilstosa(a, sv, bv):
    # Sakarto masīvu dilstoša secība izmantojot "Skaldi un valdi" algoritmu un atgriež salīdzināšanas skaitu
    # ātrums - o(nlog(n))
    # a - viendimensijas masīvs
    # sv - sākumvērtība (parasti 0)
    # bv - beigu vērtība (parasti len(a))
    p = 0
    b = numpy.arange(len(a))
    if sv < bv:
        vv = (sv + bv) // 2
        p1 = skaldi_un_valdi_dilstosa(a, sv, vv)
        p2 = skaldi_un_valdi_dilstosa(a, vv + 1, bv)
        for i in range(sv, bv + 1):
            b[i] = a[i]
        i = sv
        j = vv + 1
        k = sv
        while (i <= vv) and (j <= bv):
            p += 1
            if b[i] > b[j]:
                a[k] = b[i]
                i = i + 1
            else:
                a[k] = b[j]
                j = j + 1
            k = k + 1
        if j > bv:
```

```
        while i <= vv:
            a[k] = b[i]
            i = i + 1
            k = k + 1
        p = p + p1 + p2
    return p


def is_natural(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav
    # Ja ir naturāls skaitlis, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    if str(n).isdigit() and float(n) == int(n) and int(n) > 0:
        return True
    else:
        return False


def izvadit_masivu(x):
    # Izvada masīva elementus ar komatiem
    # x - viendimensijas masīvs
    n = len(x)
    s = str(x[0])
    for i in range(1, n):
        s = s + ", " + str(x[i])
    return s


def sort_atspole_dilstosa(a):
    # Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu
    # Kārtošanas tiek izmantota atspoles metode
```

```
    # a - viendimensijas masīvs
    n = len(a)
    count = 0
    for i in range(1, n):
        if a[i - 1] < a[i]:
            count += 1
            for j in range(i, 0, -1):
                if a[j - 1] < a[j]:
                    count += 1
                    x = a[j]
                    a[j] = a[j - 1]
                    a[j - 1] = x
                else:
                    count += 1
                    break
        else:
            count += 1

    return count



def sort_ievietosana_dilstosa(a):
    # Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu
    # Kārtošanas tiek izmantota ievietošanas metode (insertion sort)
    # a - viendimensijas masīvs
    n = len(a)
    sk = 0
    for i in range(1, n):
        sk = sk + 1
        if a[i - 1] < a[i]:
            x = a[i]
```

```
            j = i
            sk = sk + 1
            while a[j - 1] < x:
                a[j] = a[j - 1]
                j = j - 1
                if j == 0:
                    break
                sk = sk + 1
            a[j] = x

    return sk


def sort_sella_dilstosa(a):
    # Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu
    # Kārtošanas tiek izmantota Šellas metode (Shell sort)
    # a - viendimensijas masīvs
    sk = 0
    n = len(a)
    solis = (3**math.floor(math.log(2 * n + 1, 3)) - 1) // 2
    while solis >= 1:
        for k in range(0, solis):
            for i in range(solis + k, n, solis):
                sk = sk + 1
                if a[i - solis] < a[i]:
                    x = a[i]
                    j = i
                    sk = sk + 1
                    while a[j - solis] < x:
                        a[j] = a[j - solis]
                        j = j - solis
```

```
                if j == k:
                    break
                sk = sk + 1
            a[j] = x
        solis = (solis - 1) // 3

    return sk


def sort_atrais_dilstosa(a, sv, bv):
    # Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu
    # Kārtošanas tiek izmantota Hoara (ātrais) metode (quicksort)
    # a - viendimensijas masīvs
    # sv - sākuma vērtība
    # bv - beigu vērtība
    p = 0
    if sv < bv:
        i = sv
        j = bv
        solis = -1
        lv = True
        while i != j:
            if lv == (a[i] < a[j]):
                x = a[i]
                a[i] = a[j]
                a[j] = x
                x = i
                i = j
                j = x
                lv = not lv
                solis = -solis
```

```
            p += 1
            j = j + solis
        p1 = sort_atrais_dilstosa(a, sv, i - 1)
        p2 = sort_atrais_dilstosa(a, i + 1, bv)
        p = p + p1 + p2
    return p


def izvadit_masivu_un_salidzinasanas_skaitu(x, sal):
    # Izvada salīdzināšanas skaitu un izvada masīva elementus ar komatiem
    # x - viendimensijas masīvs
    # sal - int skaitlis
    n = len(x)
    s = str(x[0])
    for i in range(1, n):
        s = s + ", " + str(x[i])
    return str(sal) + " salīdzināšanas - " + s


def samaisit(masivs):
    # Samaisa masīva elementus (funkcija tiek izmantota, lai no sakārtota masīva
numpy.arange(n + 1) izveidot nesakārtotu (unsort)
    # masivs - viendimensijas masīvs
    i = 0
    while i < len(masivs) // 2:
        x = random.randint(1, len(masivs) - 1)
        y = random.randint(1, len(masivs) - 1)
        temp = masivs[x]
        masivs[x] = masivs[y]
        masivs[y] = temp
        i = i + 1
```

```
    return masivs


# ------------------------------------------------------
# Galvenā programmas daļa
# ------------------------------------------------------


n = input("Ievadiet masīva izmēru N ===> ")

while is_natural(n) == False:
    n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

n = int(n)

a = numpy.arange(n + 1)

samaisit(a)

print("\nSākotnējais masīvs:")
print(izvadit_masivu(a))

y = copy.deepcopy(a)
z = copy.deepcopy(a)
k = copy.deepcopy(a)
p = copy.deepcopy(a)
u = copy.deepcopy(a)

sorted_b = sort_atspole_dilstosa(y)
print("\nKārtošana ar atspoles metodi:")
print(izvadit_masivu_un_salidzinasanas_skaitu(y, sorted_b))
```

```
sorted_c = sort_ievietosana_dilstosa(z)
print("\nKārtošana ar ievietošanas metodi:")
print(izvadit_masivu_un_salidzinasanas_skaitu(z, sorted_c))

sorted_d = sort_sella_dilstosa(p)
print("\nKārtošana ar Šella metodi:")
print(izvadit_masivu_un_salidzinasanas_skaitu(p, sorted_d))

sorted_e = sort_atrais_dilstosa(k, 0, len(a) - 1)
print("\nKārtošana ar Hoara (ātrais) metodi:")
print(izvadit_masivu_un_salidzinasanas_skaitu(k, sorted_e))

print("\nKārtošana ar 'Skaldi un valdi' metodi:")
sorted_u = skaldi_un_valdi_dilstosa(u, 0, len(a) - 1)
print(izvadit_masivu_un_salidzinasanas_skaitu(u, sorted_u))
```

Testa piemēri:

1)

```
Ievadiet masīva izmēru N ===> 10

Sākotnējais masīvs:
0, 1, 4, 10, 8, 5, 6, 7, 2, 3, 9

Kārtošana ar atspoles metodi:
52 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar ievietošanas metodi:
52 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar Šella metodi:
49 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar Hoara (ātrais) metodi:
33 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar 'Skaldi un valdi' metodi:
25 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
```

2)

```
Ievadiet masīva izmēru N ===> 10

Sākotnējais masīvs:
0, 1, 2, 8, 4, 10, 6, 9, 3, 7, 5

Kārtošana ar atspoles metodi:
54 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar ievietošanas metodi:
54 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar Šella metodi:
51 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar Hoara (ātrais) metodi:
31 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar 'Skaldi un valdi' metodi:
23 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
```

3)

```
Ievadiet masīva izmēru N ===> 10

Sākotnējais masīvs:
0, 2, 7, 3, 10, 5, 6, 1, 8, 9, 4

Kārtošana ar atspoles metodi:
53 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar ievietošanas metodi:
53 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar Šella metodi:
39 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar Hoara (ātrais) metodi:
31 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Kārtošana ar 'Skaldi un valdi' metodi:
26 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
```

4)

```
Ievadiet masīva izmēru N ===> 1000

Sākotnējais masīvs:
0, 1, 239, 164, 4, 950, 648, 529, 782, 691, 10, 11, 438, 880, 14, 918, 22, 17, 884, 275, 20, 846, 463, 135, 286, 119, 26, 200,

Kārtošana ar atspoles metodi:
316649 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 9

Kārtošana ar ievietošanas metodi:
316649 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 9

Kārtošana ar Šella metodi:
18068 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 98

Kārtošana ar Hoara (ātrais) metodi:
12015 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 98

Kārtošana ar 'Skaldi un valdi' metodi:
8663 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 986
```

5)

```
Ievadiet masīva izmēru N ===> 1000

Sākotnējais masīvs:
0, 1, 607, 341, 4, 864, 6, 7, 160, 9, 828, 738, 496, 13

Kārtošana ar atspoles metodi:
327938 salīdzināšanas - 1000, 999, 998, 997, 996, 995,

Kārtošana ar ievietošanas metodi:
327938 salīdzināšanas - 1000, 999, 998, 997, 996, 995,

Kārtošana ar Šella metodi:
17259 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 9

Kārtošana ar Hoara (ātrais) metodi:
12959 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 9

Kārtošana ar 'Skaldi un valdi' metodi:
8608 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994
```

6)

```
Ievadiet masīva izmēru N ===> 1000

Sākotnējais masīvs:
0, 1, 2, 3, 4, 263, 6, 41, 274, 823, 202, 977, 914, 420, 908, 8, 16,

Kārtošana ar atspoles metodi:
326214 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992

Kārtošana ar ievietošanas metodi:
326214 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992

Kārtošana ar Šella metodi:
18003 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992,

Kārtošana ar Hoara (ātrais) metodi:
13814 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992,

Kārtošana ar 'Skaldi un valdi' metodi:
8593 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992,
```

7)

```
Ievadiet masīva izmēru N ===> 10000

Sākotnējais masīvs:
0, 9265, 2, 3, 4, 6858, 2287, 7, 8, 9, 10, 11, 3300, 5400,

Kārtošana ar atspoles metodi:
32279460 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9

Kārtošana ar ievietošanas metodi:
32279460 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9

Kārtošana ar Šella metodi:
276349 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 999

Kārtošana ar Hoara (ātrais) metodi:
163970 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 999

Kārtošana ar 'Skaldi un valdi' metodi:
119494 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 999
```

8)

```
Ievadiet masīva izmēru N ===> 10000

Sākotnējais masīvs:
0, 1, 2, 1610, 4, 5, 319, 7, 2505, 6362, 10, 2694, 1059, 1329

Kārtošana ar atspoles metodi:
32196310 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995

Kārtošana ar ievietošanas metodi:
32196310 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995

Kārtošana ar Šella metodi:
279824 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995,

Kārtošana ar Hoara (ātrais) metodi:
175002 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995,

Kārtošana ar 'Skaldi un valdi' metodi:
119434 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995,
```

9)

```
Ievadiet masīva izmēru N ===> 10000

Sākotnējais masīvs:
0, 220, 559, 3, 6027, 5, 6, 4499, 8, 9, 6176, 315

Kārtošana ar atspoles metodi:
31938609 salīdzināšanas - 10000, 9999, 9998, 9997,

Kārtošana ar ievietošanas metodi:
31938609 salīdzināšanas - 10000, 9999, 9998, 9997,

Kārtošana ar Šella metodi:
285344 salīdzināšanas - 10000, 9999, 9998, 9997,

Kārtošana ar Hoara (ātrais) metodi:
182775 salīdzināšanas - 10000, 9999, 9998, 9997,

Kārtošana ar 'Skaldi un valdi' metodi:
119362 salīdzināšanas - 10000, 9999, 9998, 9997,
```

10)

```
Ievadiet masīva izmēru N ===> 10000

Sākotnējais masīvs:
0, 2781, 2, 3, 4023, 5, 1341, 7, 6441, 9, 10, 2294, 2946, 5271, 4176, 2078, 9937, 4360, 2185, 6046, 20

Kārtošana ar atspoles metodi:
32271631 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 998

Kārtošana ar ievietošanas metodi:
32271631 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 998

Kārtošana ar Šella metodi:
283088 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988,

Kārtošana ar Hoara (ātrais) metodi:
167606 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988,

Kārtošana ar 'Skaldi un valdi' metodi:
119585 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988,
```

4. uzdevums

Sastādīt programmu, kas doto naturālo skaitli no 1 līdz 3899 arābu pierakstā pārveido par skaitļi romiešu pierakstā. Cipari romiešu pierakstā jāuzglabā masīvā.

Kods:

```
# Programmas nosaukums: Trīs masīvu apvienošana

# 4. uzdevums (1MPR07_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas doto naturālo skaitli no 1 līdz 3899 arābu pierakstā pārveido par skaitļi romiešu pierakstā.

# Cipari romiešu pierakstā jāuzglabā masīvā.

# Versija 1.0



import numpy



def to_roman(n):
    # Pārveido veselo skaitli no 1 līdz 3899 no arābu pieraksta uz romiešu pierakstu
    # n - naturāls skaitlis no 1 līdz 3899
    romas = numpy.array(["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", 'I'])
    vertibas = numpy.array([1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1])
    sv = ""
    for i in range(13):
        while vertibas[i] <= n:
            sv = sv + romas[i]
            n = n - vertibas[i]
    return sv



def is_natural_and_interval(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav
    # Ja ir naturāls skaitlis, tad True. Ja nav tad False.
```

```
    # n - simbolu virkne, kuru pārbauda.
    if (str(n).isdigit() and float(n) == int(n) and int(n) > 0):
        if (int(n) < 1 or int(n) > 3899):
            return False
        else:
            return True
    else:
        return False



# ---------------------------------------------------------
# Galvenā programmas daļa
# ---------------------------------------------------------



n = input("Ievadiet naturālo skaitli no 1 līdz 3899 ==> ")

while is_natural_and_interval(n) == False:

    n = input("Kļūda! \nIevadiet naturālo skaitli no 1 līdz 3899 ==> ")

n = int(n)

print("\nArābu -> Romiešu")
print(str(n) + ": " + str(to_roman(n)))
```

Testa piemēri:

1)

```
Ievadiet naturālo skaitli no 1 līdz 3899 ==> -1
Kļūda!
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 0
Kļūda!
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 1

Arābu -> Romiešu
1: I
```

2)

```
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 699

Arābu -> Romiešu
699: DCXCIX
```

3)

```
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 55

Arābu -> Romiešu
55: LV
```

4)

```
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 666

Arābu -> Romiešu
666: DCLXVI
```

5)

```
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 11

Arābu -> Romiešu
11: XI
```

6)

```
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 3900
Kļūda!
Ievadiet naturālo skaitli no 1 līdz 3899 ==> pieci
Kļūda!
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 5

Arābu -> Romiešu
5: V
```

7)

```
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 0
Kļūda!
Ievadiet naturālo skaitli no 1 līdz 3899 ==> -1212
Kļūda!
Ievadiet naturālo skaitli no 1 līdz 3899 ==> 1212

Arābu -> Romiešu
1212: MCCXII
```

PU2. uzdevums

Sastādīt programmu, kas lietotāja ievadīto reālo skaitli (< 10 000) noapaļo līdz diviem cipariem aiz komata un rezultātu nodrukā šādā formātā:

Ievade: 523.6789

Izvade: Jums ir jāmaksā EUR 523.68 (pieci simti divdesmit trīs euro un 68 euro centi)

Nepieciešami vārdi skaitļu pieraksta izveidei uzglabājami masīvā.

Kods:

Programmas nosaukums: Veikals

Papilduzdevums 2 (1MPR07_Vladislavs_Babaņins)

Uzdevuma formulējums: Sastādīt programmu, kas lietotāja ievadīto reālo skaitli (< 10 000) noapaļo līdz diviem cipariem aiz komata un rezultātu nodrukā šādā formātā:

Ievade: 523.6789

Izvade: Jums ir jāmaksā EUR 523.68 (pieci simti divdesmit trīs euro un 68 euro centi)

Nepieciešami vārdi skaitļu pieraksta izveidei uzglabājami masīvā.

Versija 1.0

```
import numpy



def to_vards(n):
    # Pārveido reālo skaitli no 0 līdz 9999 tā, ka tas ir izrūnāms latviešu valodā
    # n - reāls skaitlis no 0 līdz 9999
    lst_word = numpy.array(["desmit tūkstoši ", "deviņi tūkstoši ", "astoņi tūkstoši ", "septiņi tūkstoši ", "seši tūkstoši ", "pieci tūkstoši ", "četri tūkstoši ", "trīs tūkstoši ", "divi tūkstoši ", "tūkstošs ",
                           "deviņi simti ", "astoņi simti ", "septiņi simti ", "seši simti ", "pieci simti ", "četri simti ", "trīs simti ", "divi simti ", "simts ",
                           "deviņdesmit ", "astoņdesmit ", "septiņdesmit ", "sešdesmit ", "piecdesmit ", "četrdesmit ", "trīsdesmit ", "divdesmit ",
                           "deviņpadsmit ", "astoņpadsmit ", "septiņpadsmit ", "sešpadsmit ", "piecpadsmit ", "četrpadsmit ", "trīspadsmit ", "divpadsmit ", "vienpadsmit ", "desmit ",
                           "deviņi ", "astoņi ", "septiņi ", "seši ", "pieci ", "četri ", "trīs ", "divi ", "viens "])
```

```
    lst_numbers = numpy.array([10000, 9000, 8000, 7000, 6000, 5000, 4000, 3000, 2000, 1000,
                               900, 800, 700, 600, 500, 400, 300, 200, 100,
                               90, 80, 70, 60, 50, 40, 30, 20,
                               19, 18, 17, 16, 15, 14, 13, 12, 11, 10,
                               9, 8, 7, 6, 5, 4, 3, 2, 1])

    sv = ""
    for i in range(46):
        while lst_numbers[i] <= n:
            sv = sv + lst_word[i]
            n = n - lst_numbers[i]

    if sv == "":
        return "nulle "
    else:
        return sv


def check():
    # Bezgalīgi daudz ievades. Pārbauda vai skaitlis ir reāls pozitīvs un atrodas intervālā [0, 9999]
    x = input("Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> ")
    while True:
        try:
            x = float(x)
        except:
            x = input("Kļūda!\nIevadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> ")
        else:
            if x < 0 or x > 9999:
                x = input("Kļūda! Skaitlis nepieder intervālam [0, 9999]\nIevadiet reālo pozitīvo skaitli
no 0 līdz 9999 ===> ")
```

```
        else:
            return float(x)


# ---------------------------------------------------------
# Galvenā programmas daļa
# ---------------------------------------------------------


input_number = check()


input_number = round(input_number, 2)


integer_part = int(input_number)
decimal_part = int(round((input_number - integer_part) * 100))


print("\nJums ir jāmaksā EUR " + str(input_number) + " (" + str(to_vards(integer_part) + "euro un " + str(decimal_part) + " euro centi)"))
```

Testa piemēri:

1)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 523.6789

Jums ir jāmaksā EUR 523.68 (pieci simti divdesmit trīs euro un 68 euro centi)
```

2)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 16165
Kļūda! Skaitlis nepieder intervālam [0, 9999]
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> -12
Kļūda! Skaitlis nepieder intervālam [0, 9999]
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> pieci
Kļūda!
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 15

Jums ir jāmaksā EUR 15.0 (piecpadsmit euro un 0 euro centi)
```

3)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 16.1689441631

Jums ir jāmaksā EUR 16.17 (sešpadsmit euro un 17 euro centi)
```

4)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 9999

Jums ir jāmaksā EUR 9999.0 (deviņi tūkstoši deviņi simti deviņdesmit deviņi euro un 0 euro centi)
```

5)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 6666

Jums ir jāmaksā EUR 6666.0 (seši tūkstoši seši simti sešdesmit seši euro un 0 euro centi)
```

6)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 666

Jums ir jāmaksā EUR 666.0 (seši simti sešdesmit seši euro un 0 euro centi)
```

7)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 0

Jums ir jāmaksā EUR 0.0 (nulle euro un 0 euro centi)
```

8)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 12.464654654165614653

Jums ir jāmaksā EUR 12.46 (divpadsmit euro un 46 euro centi)
```

9)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 4612.89

Jums ir jāmaksā EUR 4612.89 (četri tūkstoši seši simti divpadsmit euro un 89 euro centi)
```

10)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 10000
Kļūda! Skaitlis nepieder intervālam [0, 9999]
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 1

Jums ir jāmaksā EUR 1.0 (viens euro un 0 euro centi)
```

11)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 1234

Jums ir jāmaksā EUR 1234.0 (tūkstošs divi simti trīsdesmit četri euro un 0 euro centi)
```

12)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 9876

Jums ir jāmaksā EUR 9876.0 (deviņi tūkstoši astoņi simti septiņdesmit seši euro un 0 euro centi)
```

13)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 1111.1111111111

Jums ir jāmaksā EUR 1111.11 (tūkstošs simts vienpadsmit euro un 11 euro centi)
```

14)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 2222.22

Jums ir jāmaksā EUR 2222.22 (divi tūkstoši divi simti divdesmit divi euro un 22 euro centi)
```

15)

```
Ievadiet reālo pozitīvo skaitli no 0 līdz 9999 ===> 13.445

Jums ir jāmaksā EUR 13.45 (trīspadsmit euro un 45 euro centi)
```

5. uzdevums

Jums ir jāizveido viendimensijas masīvs ar 10 000 elementu. Šī masīva elementi ir uz labu laimi ģenerētas simbolu virknes (vārdi) garumā no 3 līdz 8 simboliem un sastāv tikai no lielajiem latīņu alfabēta burtiem. Pēc šī masīva izveides Jums jāveic visas nepieciešamās darbības, lai varētu ātri (ne sliktāk kā logaritmiskā laikā) pārbaudīt, vai šis masīvs satur vai nesatur lietotāja ievadīto frāzi.

Kods:

```
# Programmas nosaukums: Pārbauda vai masīvs satur frāzi

# 5. uzdevums (1MPR07_Vladislavs_Babaņins)

# Uzdevuma formulējums: Jums ir jāizveido viendimensijas masīvs ar 10 000 elementu.

# Šī masīva elementi ir uz labu laimi ģenerētas simbolu virknes (vārdi) garumā no 3 līdz 8 simboliem un sastāv tikai no

# lielajiem latīņu alfabēta burtiem. Pēc šī masīva izveides Jums jāveic visas nepieciešamās darbības, lai varētu ātri (ne sliktāk kā logaritmiskā laikā)

# pārbaudīt, vai šis masīvs satur vai nesatur lietotāja ievadīto frāzi.

# Versija 1.0


import numpy

import random
```

```
def atrais(a, sv, bv):  # sv = 0, bv = len(a)
    # Sakārto masīvu augoša secība
    # Kārtošanas tiek izmantota Hoara (ātrais) metode (quicksort)
    # a - viendimensijas masīvs
    # sv - sākuma vērtība
    # bv - beigu vērtība
    if sv < bv:
        i = sv
        j = bv
        solis = -1
        lv = True
        while i != j:
            if lv == (a[i] > a[j]):
                x = a[i]
                a[i] = a[j]
                a[j] = x
                x = i
                i = j
                j = x
                lv = not lv
                solis = -solis
            j = j + solis
        atrais(a, sv, i - 1)
        atrais(a, i + 1, bv)


def vards():
    # Ģenerē vārdus ar garumu no 3 līdz 8 (garums - uz labu laimi) un ģenerē to vārdu ar lieliem latiņu alfabēta burtiem
    # Atgriež vienu izveidotu vārdu
```

```
    r = random.randint(3, 8)
    v = ""
    for i in range(r):
        v += chr(random.randint(65, 90))  # ASCII 65;90
    return v


def masivs(length):
    # Aizpildā masīvu ar vārdiem atsaucoties uz "vards()". Atgriež aizpildīto masīvu.
    # length - viendimensijas masīva garums
    mas = numpy.empty(length, dtype=object)
    for j in range(length):
        mas[j] = vards()
    # print(mas)
    return mas


def meklet(a, b):
    # Sameklē a masīva b skaitļi (vai vārdu). Atgriež to vērtību, kur viņa atrodas pēc index. Ja nav tādas vērtības masīva, tad atgriež -1.
    # a - viendimensijas masīvs
    # b - to ko mēs meklējam (skaitlis vai vārds (str))
    l = 0
    r = len(a) - 1
    while (l <= r):
        i = (l + r) // 2
        if a[i] == b:
            break
        elif a[i] < b:
            l = i + 1
        else:
```

```
            r = i - 1
    if a[i] == b:
        return i
    else:
        return -1


def izvade(x):
    # Izvada masīva elementus pēc kārtas līdz pedējam
    # x - viendimensijas masīvs
    n = len(x)
    s = str(x[0])
    for i in range(1, n):
        s = s + ", " + str(x[i])
    print(s)


# ------------------------------------------------------
# Galvenā programmas daļa
# ------------------------------------------------------


lenght = 10000
a = masivs(lenght)

atrais(a, 0, len(a) - 1)

print("Uz labu laimi izveidotie vārdi:")
izvade(a)
print("")
```

```
m = input("Ievadi meklējamo ===> ")


vieta = meklet(a, m)


if vieta == -1:
    print("Meklējamais vārds " + str(m) + " nav atrasts.")
else:
    print("Meklējamais ir " + str(vieta) + ". vietā.")
```

Testa piemēri:

1)

```
Uz labu laimi izveidotie vārdi:
AAB, AABFEMLO, AABRW, AADLDMP, AAGRL, AAHK, AAHLOS,

Ievadi meklējamo ===> AAB
Meklējamais ir 0. vietā.
```

2)

```
Uz labu laimi izveidotie vārdi:
AABFT, AABFYX, AACYKBD, AAE, AAJHJE, AAM, AAMX, AAN, AAQBHF,

Ievadi meklējamo ===> AAN
Meklējamais ir 7. vietā.
```

3)

```
Uz labu laimi izveidotie vārdi:
AAAMOYT, AABFO, AABTGA, AAHXQFX, AAJIYEAK, AAKR, AAL, AAMH, AAMV, AANGY, AARAJJ, AATQYG,

Ievadi meklējamo ===> PIECI
Meklējamais vārds PIECI nav atrasts.
```

4)

```
Uz labu laimi izveidotie vārdi:
AACNTOA, AAHECTF, AAHRU, AAHS, AAMT, AANG, AAOEAG, AAR, AARHQTU, AARODNJ, AARONTPQ, AAS, AAU,

Ievadi meklējamo ===> ZZY
Meklējamais ir 9997. vietā.
```

5)

```
Uz labu laimi izveidotie vārdi:
AAAE, AAB, AABGJEME, AABN, AACSRQI, AACW, AAIHVNWM, AALV, AAM, AAMRHKSG, AAO, AAO, AAPJGVI,

Ievadi meklējamo ===> MXEPAV
Meklējamais ir 5008. vietā.
```

6)

```
Uz labu laimi izveidotie vārdi:
AAB, AABRK, AAG, AAMVIB, AANOL, AAOVX, AAP, AAPKWURQ,

Ievadi meklējamo ===> OREO
Meklējamais vārds OREO nav atrasts.
```

7)

```
Uz labu laimi izveidotie vārdi:
AAADYJ, AAD, AAETEAAX, AAFJ, AAFSU, AAFVV, AAH, AAHSIH,

Ievadi meklējamo ===> 5
Meklējamais vārds 5 nav atrasts.
```

8)

```
Uz labu laimi izveidotie vārdi:
AAAL, AABW, AAC, AADNTIWS, AADP, AADUGY, AAEP, AAFOTB, AAHP, AAIXWRPA,

Ievadi meklējamo ===> LATVIJ
Meklējamais vārds LATVIJ nav atrasts.
```

9)

```
Uz labu laimi izveidotie vārdi:
AAAGYAJ, AACLTHH, AACPYO, AADNONRR, AAEKU, AAF, AAIBS, AAINYUQ, AAKQUGC, AANJZN, AAORA,

Ievadi meklējamo ===> AAINYUQ
Meklējamais ir 7. vietā.
```

10)

```
Uz labu laimi izveidotie vārdi:
AACLQUKX, AADCI, AADDNN, AADKQ, AADTHOQ, AADX, AAE, AAGE, AAGGZRR, AAI, AAKEUHHN, AAKPV, AALA, AAMSUJ,

Ievadi meklējamo ===> PSG
Meklējamais ir 6002. vietā.
```