6. praktiskais darbs. 2. semestris

1. uzdevums

Realizēt 4 kārtošanas algoritmus, kas sakārto masīva elementus dilstošā (neaugošā) secībā (5 punkti):

1.1. Atspoles metode
1.2. Ievietošanas metode
1.3. Šella metode
1.4. Hoara jeb ātrās kārtošanas metode

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

1. Uzdevumu izpildes nosacījumi:
   - Uzsākot programmas izpildi, tiek izveidots masīvs, kas satur N + 1 elementu, kur skaitli N ievada lietotājs.
   - Tiek izveidota funkcija, kas aizpilda šo masīvu pēc šādiem nosacījumiem: masīva elements ar indeksu 0 saņem vērtību 0, bet pārējie masīva elementi – gadījuma naturālos skaitļus intervālā no 1 līdz N, paredzot, ka tie masīvā neatkārtosies.
   - Izveidot funkciju, kas doto masīvu pārvērš par simbolu virkni, kuras sākumā norādīts salīdzināšanas skaits, kam seko sakārtotie masīva elementi, kuri viens no otra ir atdalīti ar komatu, piemēram, ***17 salīdzināšanas – 7, 6, 5, 4, 3, 2, 1,0***
   - Izveidotas četras funkcijas (procedūras), kas dotā masīva elementus sakārto dilstošā (neaugošā) un atgriež masīva elementu salīdzināšanas reižu skaits, kāds bija, realizējot šo kārtošanas metodi.
   - Visas četras kārtošanas metodes tiek pielietotas vienam un tam pašam sākotnēji izveidotajam un aizpildītajam masīvam.
   - Programmas izpildes rezultātā uz ekrāna tiek parādīti visi četri kārtošanas metožu izpildes rezultāti, sakārtojot 10, 1 000 un 100 000 skaitļus.

Kods:

```
# Programmas nosaukums: Četras kārtošanas algoritmi

# 1. uzdevums (1MPR06_Vladislavs_Babaņins)

# Uzdevuma formulējums: Realizēt 4 kārtošanas algoritmus, kas sakārto masīva elementus dilstošā (neaugošā) secībā (5 punkti):

# 1.1. Atspoles metode

# 1.2. Ievietošanas metode

# 1.3. Šella metode

# 1.4. Hoara jeb ātrās kārtošanas metode
```

```
    # NB! Nedrīkst rakstīt algoritmu, kas vispirms sakārto masīvu augošā secībā un pēc tam to nodrukā apgrieztā secībā!
    # Programmas autors: Vladislavs Babaņins
    # Versija 1.0


    import numpy
    import random
    import math


    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 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 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 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 = numpy.copy(a)
z = numpy.copy(a)
k = numpy.copy(a)
p = numpy.copy(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))
```

Testa piemēri:

1)

```
Ievadiet masīva izmēru N ===> 10

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

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

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

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

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

2)

```
Ievadiet masīva izmēru N ===> 10

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

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

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

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

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

3)

```
Ievadiet masīva izmēru N ===> 10

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

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

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

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

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

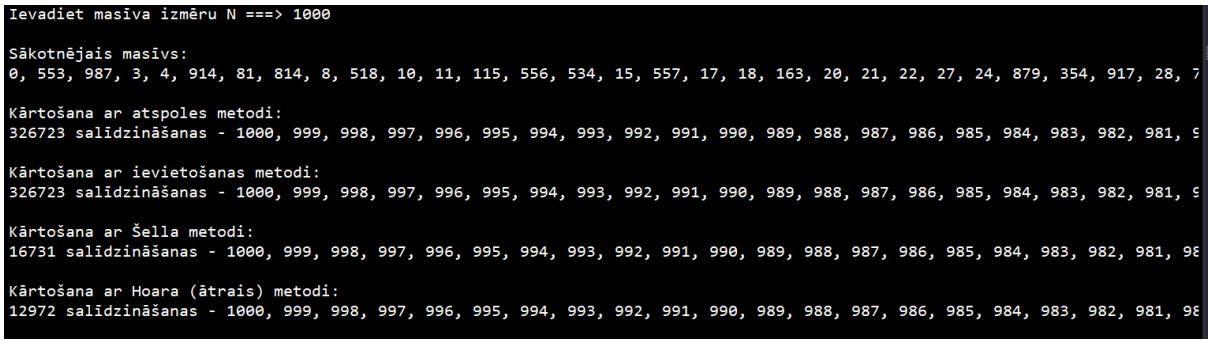
4)

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

Sākotnējais masīvs:
0, 362, 420, 492, 4, 5, 6, 392, 488, 210, 10, 897, 846, 752, 14, 271, 16, 412, 975, 643, 829, 1000, 22, 23, 24, 25, 26, 613, 3

Kārtošana ar atspoles metodi:
316966 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:
316966 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:
17937 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:
12407 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 98
```

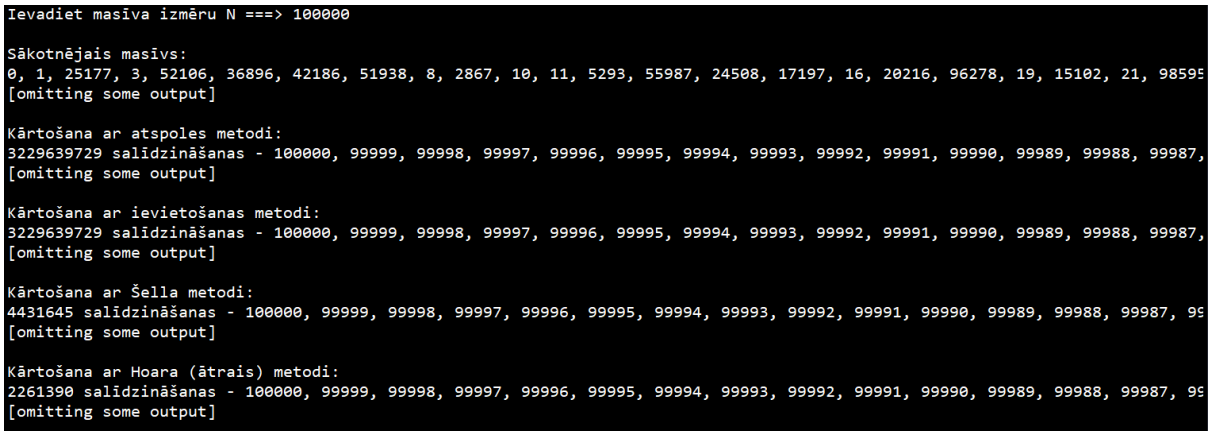
5)

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

Sākotnējais masīvs:
0, 1, 2, 3, 658, 621, 311, 480, 8, 9, 166, 463, 12, 217, 849, 176, 16, 495, 525, 927, 431, 21, 130, 652, 24, 620, 26, 945, 412

Kārtošana ar atspoles metodi:
327320 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:
327320 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:
16992 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:
14117 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 98
```

6)



7)



2. uzdevums

Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu. Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas mediānu. (1 punkts)

Kods:

Programmas nosaukums: Kopas mediāna

2. uzdevums (1MPR06_Vladislavs_Babaņins)

Uzdevuma formulējums: Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu. Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas mediānu. (1 punkts)

Programmas autors: Vladislavs Babaņins

Versija 1.0

```
import numpy

import math
```

```
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 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 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 sort_atrais_augosa(a, sv, bv):
    # Sakārto masīvu augoša secība un atgriež salīdzināšanas skaitu, lai sakartotu masīvu
    # 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
        sort_atrais_augosa(a, sv, i - 1)
        sort_atrais_augosa(a, i + 1, bv)
```

```
def is_even_masivs(x):
    # Pārbauda vai masīva garums ir pāra skaitlis
    # Ja masīva garums ir pāra skaitlis, tad atgriež True
    # Ja masīva garums nav pāra skaitlis, tad atgriež False
    # x - viendimensijas masīvs
    masiva_garums = len(x)
    if masiva_garums % 2 == 0:
        return True
    else:
        return False


def mediana(x):
    # Atgriež masīva x mediānu
    # x - viendimensijas masīvs
    if is_even_masivs(x):
        return (x[len(x) // 2 - 1] + x[len(x) // 2]) / 2
    else:
        return x[len(x) // 2]


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
# ------------------------------------------------------------


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

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

m = int(m)

print("Ievadiet masīva skaitļus!")
t = izveidot_masivu_ar_garumu(m)

print("\nIevadīta skaitļu kopa:")
izvade(t)

sort_atrais_augosa(t, 0, len(t) - 1)
print("\nSakārtota skaitļu kopa:")
izvade(t)

print("\nKopas mediāna ir:\n" + str(mediana(t)))
```

Testa piemēri:

1)

```
Ievadiet masīva izmēru N ===> 5
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 3
Ievadiet 1.elementu ===> 54
Ievadiet 2.elementu ===> 1
Ievadiet 3.elementu ===> 2
Ievadiet 4.elementu ===> 9

Ievadīta skaitļu kopa:
3, 54, 1, 2, 9

Sakārtota skaitļu kopa:
1, 2, 3, 9, 54

Kopas mediāna ir:
3
```

2)

```
Ievadiet masīva izmēru N ===> 6
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 3
Ievadiet 2.elementu ===> 2
Ievadiet 3.elementu ===> 6
Ievadiet 4.elementu ===> 8
Ievadiet 5.elementu ===> 10

Ievadīta skaitļu kopa:
1, 3, 2, 6, 8, 10

Sakārtota skaitļu kopa:
1, 2, 3, 6, 8, 10

Kopas mediāna ir:
4.5
```

3)

```
Ievadiet masīva izmēru N ===> 10
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3
Ievadiet 3.elementu ===> 4
Ievadiet 4.elementu ===> 5
Ievadiet 5.elementu ===> 6
Ievadiet 6.elementu ===> 7
Ievadiet 7.elementu ===> 8
Ievadiet 8.elementu ===> 9
Ievadiet 9.elementu ===> 10

Ievadīta skaitļu kopa:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Sakārtota skaitļu kopa:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Kopas mediāna ir:
5.5
```

4)

```
Ievadiet masīva izmēru N ===> 1
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1

Ievadīta skaitļu kopa:
1

Sakārtota skaitļu kopa:
1

Kopas mediāna ir:
1
```

5)

```
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 masīva skaitļus!
Ievadiet 0.elementu ===> pieci
Kļūda! Ievadiet 0.elementu ===> 5
Ievadiet 1.elementu ===> 0
Ievadiet 2.elementu ===> -6
Ievadiet 3.elementu ===> 999
Ievadiet 4.elementu ===> 2

Ievadīta skaitļu kopa:
5, 0, -6, 999, 2

Sakārtota skaitļu kopa:
-6, 0, 2, 5, 999

Kopas mediāna ir:
2
```

6)

```
Ievadiet masīva izmēru N ===> 2
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2

Ievadīta skaitļu kopa:
1, 2

Sakārtota skaitļu kopa:
1, 2

Kopas mediāna ir:
1.5
```

7)

```
Ievadiet masīva izmēru N ===> 7
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1.6
Kļūda! Ievadiet 0.elementu ===> 6
Ievadiet 1.elementu ===> 1
Ievadiet 2.elementu ===> 88
Ievadiet 3.elementu ===> 13-
Kļūda! Ievadiet 3.elementu ===> -13
Ievadiet 4.elementu ===> .0
Kļūda! Ievadiet 4.elementu ===> 0
Ievadiet 5.elementu ===> -7
Ievadiet 6.elementu ===> 6666666

Ievadīta skaitļu kopa:
6, 1, 88, -13, 0, -7, 6666666

Sakārtota skaitļu kopa:
-13, -7, 0, 1, 6, 88, 6666666

Kopas mediāna ir:
1
```

PU1. uzdevums

Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu. Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas modu. Ja ir vairākas modas, tad jāpaziņo visas, bet, ja modas nav (t.i. visas atšķirīgas vērtības vienādā skaitā), tad jāpaziņo “moda netika atrasta” (2 punkti).

Kods:

```
# Programmas nosaukums: Modas noteikšana

# Papilduzdevums 1 (1MPR06_Vladislavs_Babaņins)

# Uzdevuma formulējums: Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu.

# Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas modu. Ja ir vairākas modas, tad jāpaziņo visas, bet, ja modas nav

# (t.i. visas atšķirīgas vērtības vienādā skaitā), tad jāpaziņo “moda netika atrasta” (2 punkti).

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import numpy



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 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 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 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 mode(masivs):
    # Atrod masīva modu un atgriež masīvu ar modam (vai modu)
    # Ja masīva visi elementi ir vienādi, tad atgriež False (moda netika atrasta)
    # masivs - viendimensijas masīvs (kurš vel netika kārtots augoša secība)
    '''
    Paskaidrojums, kā šī funkcija darbojas:

    Funkcija vispirms paņem, ka moda ir pirmais sakārtotā masīva elements.
    Pēc tam, funkcija iziet cauri katram masīva elementam un atjaunina sarakstu ar modam un to biežumu katru reizi, kad tiek atrasta jauna mode ar augstāku biežumu (elements, kuru ir vairāk).
    Ja tas saskaras ar modu ar tādu pašu biežumu (biežums - skaits, cik reizes paradās šis elements) kā pašreizēja moda, tad tas pievieno jauno modu, modu sarakstam.

    Funkcija iterē katru elementu sakārtotajā masīvā un pārbauda, vai tas ir vienāds ar iepriekšējo elementu.
    Ja tā ir, mainīgais "sk" tiek palielināts. Ja tā nav, funkcija pārbauda, vai pašreizējais skaits ir lielāks par līdz šim saglabāto maksimālo skaitu.
    Ja ir lielāks, tad maksimālais skaits tiek atjaunināts un pašreizējais skaitlis (elements) tiek pievienots modas sarakstam.
    Ja pašreizējais skaits ir vienāds ar maksimālo līdz šim saglabāto skaitu, modas sarakstam tiek pievienots arī šis skaitlis (elements) (vairākas modes).
    Pēc tam kad tika "iziterēts" viss masīvs, tad pārbaudām, vai pēdējā elementa biežums ir lielāks par līdz šim redzēto maksimālo biežumu.
    Ja tā ir, tad maksimālais skaits tiek atjaunināts un modas saraksts ir tas pēdējais elements.
    Ja pēdējā elementa skaits ir vienāds ar maksimālo līdz šim redzēto (saglabāto) skaitu, modas sarakstam tiek pievienots arī pēdējais elements (vairākas modes).

    Ja maksimālais biežums ir vienāds ar 1 (t.i., visi masīva elementi ir unikāli), funkcija atgriež vērtību False.
    Pretējā gadījumā tas atgriež atrasto modas sarakstu.
    '''
```

```
    sort_atrais_augosa(masivs, 0, len(masivs) - 1)  # Ievadītais masīvs tiek izkārtots augošā secībā, izmantojot sort_atrais_augosa
    sk = 1  # seko pašreizējā apstrādājamā skaitļa (elementa) biežumam
    max_sk = 1  # saglabā jebkura masīva skaitļa (elementa) maksimālo biežumu
    modas_list = [masivs[0]]  # līdz šim atrasto modu saraksts

    if len(masivs) == 1:  # Ja masīva garums ir 1, tad moda arī ir tas skaitlis (vienīgais elements)
        modas_list = [masivs[0]]
        return modas_list

    for i in range(1, len(masivs)):
        if masivs[i] == masivs[i - 1]:  # Ja elements i ir vienāds ir i - 1, tad palielinām skaitītāju
            sk = sk + 1
        else:
            if sk > max_sk:  # ja skaititajs kļūst lielāks nekā maksimālais skaitītājs
                max_sk = sk  # tad maksimālais skaititajs atjaunojas
                modas_list = [masivs[i - 1]]  # modas saraksts atjaunojas un paliek tas masīva elements, kurš vislielāko reizi paradījās pēc skaitītajā

            elif sk == max_sk:  # Ja skaititajs ir vienāds ar pašreizējo maksimālo skaitītāju (max biežumu)
                modas_list.append(masivs[i - 1])  # tad pievienojam sarakstam jauno modu (papildus modu)

            sk = 1  # pēc iterācijām sk atkal kļūst par 1

    if sk > max_sk:  # Tas ir veidots pēdējam elementam. Ja lielāks skaititajs nekā pašreizējais maksimālais.
        max_sk = sk  # tad atjaunojam maksimālo skaitītāju
        modas_list = [masivs[-1]]  # atjaunojam modas sarakstu

    elif sk == max_sk:  # Ja modas biežumi ir vienādi, tad pievienojam modas sarakstam
        modas_list.append(masivs[-1])
```

```
        if max_sk == 1:  # Ja maksimālais biežums ir vienāds ar 1 (visi masīva elementi ir unikāli), atgriež vērtību False
            return False
        else:
            return modas_list  # Atgriež atrasto modas sarakstu



    def sort_atrais_augosa(a, sv, bv):
        # Sakārto masīvu augoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu
        # 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
            sort_atrais_augosa(a, sv, i - 1)
```

```
        sort_atrais_augosa(a, i + 1, bv)


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



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


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


m = int(m)


print("Ievadiet masīva skaitļus!")
t = izveidot_masivu_ar_garumu(m)


print("\nIevadīta skaitļu kopa:")
izvade(t)


sort_atrais_augosa(t, 0, len(t) - 1)
print("\nSakārtota skaitļu kopa:")
izvade(t)


modas = mode(t)


if modas == False:
    print("\nModa netika atrasta.")
else:
    print("\nSkaitļu kopas moda(s) ir:")
    izvade(modas)
```

Testa piemēri:

1)

```
Ievadiet masīva izmēru N ===> 3
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3

Ievadīta skaitļu kopa:
1, 2, 3

Sakārtota skaitļu kopa:
1, 2, 3

Moda netika atrasta.
```

2)

```
Ievadiet masīva izmēru N ===> 5
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 87
Ievadiet 1.elementu ===> 5
Ievadiet 2.elementu ===> 66
Ievadiet 3.elementu ===> 2
Ievadiet 4.elementu ===> 1

Ievadīta skaitļu kopa:
87, 5, 66, 2, 1

Sakārtota skaitļu kopa:
1, 2, 5, 66, 87

Moda netika atrasta.
```

3)

```
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 masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 1
Ievadiet 2.elementu ===> 1
Ievadiet 3.elementu ===> 1
Ievadiet 4.elementu ===> 1

Ievadīta skaitļu kopa:
1, 1, 1, 1, 1

Sakārtota skaitļu kopa:
1, 1, 1, 1, 1

Skaitļu kopas moda(s) ir:
1
```

4)

```
Ievadiet masīva izmēru N ===> 6
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 5
Ievadiet 1.elementu ===> 3
Ievadiet 2.elementu ===> 5
Ievadiet 3.elementu ===> 6
Ievadiet 4.elementu ===> 6
Ievadiet 5.elementu ===> 3

Ievadīta skaitļu kopa:
5, 3, 5, 6, 6, 3

Sakārtota skaitļu kopa:
3, 3, 5, 5, 6, 6

Skaitļu kopas moda(s) ir:
3, 5, 6
```

5)

```
Ievadiet masīva izmēru N ===> 10
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 12
Ievadiet 1.elementu ===> 515
Ievadiet 2.elementu ===> pieci
Kļūda! Ievadiet 2.elementu ===> 15512
Ievadiet 3.elementu ===> 2423
Ievadiet 4.elementu ===> 6236
Ievadiet 5.elementu ===> 231
Ievadiet 6.elementu ===> 222
Ievadiet 7.elementu ===> 1
Ievadiet 8.elementu ===> 1
Ievadiet 9.elementu ===> 2

Ievadīta skaitļu kopa:
12, 515, 15512, 2423, 6236, 231, 222, 1, 1, 2

Sakārtota skaitļu kopa:
1, 1, 2, 12, 222, 231, 515, 2423, 6236, 15512

Skaitļu kopas moda(s) ir:
1
```

6)

```
Ievadiet masīva izmēru N ===> 10
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 1
Ievadiet 2.elementu ===> 2
Ievadiet 3.elementu ===> 2
Ievadiet 4.elementu ===> 3
Ievadiet 5.elementu ===> 3
Ievadiet 6.elementu ===> 4
Ievadiet 7.elementu ===> 4
Ievadiet 8.elementu ===> 5
Ievadiet 9.elementu ===> 5

Ievadīta skaitļu kopa:
1, 1, 2, 2, 3, 3, 4, 4, 5, 5

Sakārtota skaitļu kopa:
1, 1, 2, 2, 3, 3, 4, 4, 5, 5

Skaitļu kopas moda(s) ir:
1, 2, 3, 4, 5
```

7)

```
Ievadiet masīva izmēru N ===> 10
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> -1
Ievadiet 1.elementu ===> -1
Ievadiet 2.elementu ===> 2
Ievadiet 3.elementu ===> 2
Ievadiet 4.elementu ===> 244
Ievadiet 5.elementu ===> 244
Ievadiet 6.elementu ===> 33
Ievadiet 7.elementu ===> 33
Ievadiet 8.elementu ===> 55
Ievadiet 9.elementu ===> 5

Ievadīta skaitļu kopa:
-1, -1, 2, 2, 244, 244, 33, 33, 55, 5

Sakārtota skaitļu kopa:
-1, -1, 2, 2, 5, 33, 33, 55, 244, 244

Skaitļu kopas moda(s) ir:
-1, 2, 33, 244
```

8)

```
Ievadiet masīva izmēru N ===> 6
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 1
Ievadiet 2.elementu ===> 1
Ievadiet 3.elementu ===> 3
Ievadiet 4.elementu ===> 3
Ievadiet 5.elementu ===> 3

Ievadīta skaitļu kopa:
1, 1, 1, 3, 3, 3

Sakārtota skaitļu kopa:
1, 1, 1, 3, 3, 3

Skaitļu kopas moda(s) ir:
1, 3
```

9)

```
Ievadiet masīva izmēru N ===> 10
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 3
Ievadiet 3.elementu ===> 4
Ievadiet 4.elementu ===> 5
Ievadiet 5.elementu ===> 6
Ievadiet 6.elementu ===> 7
Ievadiet 7.elementu ===> 8
Ievadiet 8.elementu ===> 9
Ievadiet 9.elementu ===> 10

Ievadīta skaitļu kopa:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Sakārtota skaitļu kopa:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Moda netika atrasta.
```

10)

```
Ievadiet masīva izmēru N ===> 1
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 0

Ievadīta skaitļu kopa:
0

Sakārtota skaitļu kopa:
0

Skaitļu kopas moda(s) ir:
0
```

11)

```
Ievadiet masīva izmēru N ===> 0
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> -2
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> viens
Masīva izmērs ir naturāls skaitlis!
Ievadiet masīva izmēru N ===> 1
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> pieci
Kļūda! Ievadiet 0.elementu ===> 6

Ievadīta skaitļu kopa:
6

Sakārtota skaitļu kopa:
6

Skaitļu kopas moda(s) ir:
6
```

12)

```
Ievadiet masīva izmēru N ===> 5
Ievadiet masīva skaitļus!
Ievadiet 0.elementu ===> 1
Ievadiet 1.elementu ===> 2
Ievadiet 2.elementu ===> 4
Ievadiet 3.elementu ===> 7
Ievadiet 4.elementu ===> 0

Ievadīta skaitļu kopa:
1, 2, 4, 7, 0

Sakārtota skaitļu kopa:
0, 1, 2, 4, 7

Moda netika atrasta.
```