8. praktiskais darbs. 2. semestris

1. uzdevums

Sastādīt programmu, kas izveido visu pirmskaitļu, kas nepārsniedz 10000, augošā secībā sakārtotu masīvu un izvada šo masīvu uz ekrāna. Atrast un pielietot formulu, kas nosaka iespējamo pirmskaitļu, kas nepārsniedz patvaļīgu skaitli N, skaitu, definējot pirmskaitļu masīva garumu.

Kods:

```
# Programmas nosaukums: Izveido masīvu ar visiem pirmskaitļiem, kuri ir ne lielāki 10000 un izvada to augošā secībā

# 1. uzdevums (1MPR08_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas izveido visu pirmskaitļu, kas nepārsniedz 10000, augošā secībā sakārtotu masīvu

# un izvada šo masīvu uz ekrāna. Atrast un pielietot formulu, kas nosaka iespējamo pirmskaitļu, kas nepārsniedz patvaļīgu skaitli N,

# skaitu, definējot pirmskaitļu masīva garumu.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import numpy

import math



def izvade(x):

    # Izvada masīva elementus pēc kārtas līdz pedējam

    # x - viendimensijas masīvs


    try:

        n = len(x)

        s = str(x[0])

        for i in range(1, n):
```

```
            s = s + ", " + str(x[i])
        print(s)
    except IndexError:
        print("Nav pirmskaitļu šajā intervālā.")  # Ja parādās index error, tad tas nozīme, ka šajā intervālā nav pirmskaitļu.
    except:
        print("Kļūda!")  # Citām kļūdam


def pirmskaitlu_masivs(n, drosibas_koeficients):
    # Atgriež tuple ar pirmskaitļu masīvu līdz skaitlim n bez liekām 0 un ar nodzēsto nulles skaitu. (Ja ir lieki elementi, tad funkcija to nodzes)
    # n - naturāls skaitlis (int), līdz kurām meklēsīm pirmskaitļus
    # drosibas_koeficients - drošības koeficients. Tiek reizināts ar n/log(n) un tiek izmantots masīva izmēra definēšanai.
    # Parasti, jo lielāks, jo vairāk liekas nulles būs. Pēc noklusējuma labāk, lai tās būtu aptuvēni 1.2

    if n <= 1:  # Funkcija nedarbotos jā ievadīs 0, 1 vai 2, tāpēc tas tiek atsevišķi definēts.
        return numpy.array([])
    if n == 2:
        return numpy.array([2])

    # drosibas_koeficients = 1.2  # izmantots, lai palidzētu nodefinētu masīva izmēru (size). Masīva izmēra definēšanai tiek izmantota formula n/log(n) https://en.wikipedia.org/wiki/Prime_number_theorem un https://en.wikipedia.org/wiki/Prime-counting_function
    size = math.ceil((n / math.log(n)) * drosibas_koeficients)  # aprēķina masīva garumu, izmantojot formulu n/log(n) un reizinot ar drošības koeficientu (bez viņa nevienmēr pietiks vietas)
    p = numpy.zeros(size, dtype=numpy.int32)  # izveido masīvu ar 0 vērtībām tādu garumu, kuru aprēķinaja ar formulu (n/log(n))*drosibas_koeficientu
    p[0] = 2  # pirmais pirmskaitlis
    p[1] = 3  # otrais pirmskaitlis
    j = 2
```

```
    k = 5
    try:
        while k <= n:
            i = 0
            s = round(math.sqrt(k))
            while (k % p[i]) != 0:
                i = i + 1
                if p[i] > s:
                    p[j] = k
                    j = j + 1
                    break
            k = k + 2
        deleted_zeros_count = len(p) - len(p[:j])
        return (p[:j], deleted_zeros_count)  # [:j], lai izvadītu bez nullem
    except:
        print("Palieliniet drošības koeficientu! Nepietika vietas masīva aizpildīšanai.")



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



n = 10000  # Līdz kuram skaitlim meklējam pirmskaitļus
drosibas_koeficients = 1.2  # izmantots, lai palidzētu nodefinētu masīva izmēru (size). Masīva izmēra definēšanai tiek izmantota formula n/log(n) https://en.wikipedia.org/wiki/Prime_number_theorem un https://en.wikipedia.org/wiki/Prime-counting_function
print("Pimskaitļi līdz " + str(n) + ":")  # Var mainīt drošības koeficientu, lai mainītu definētā masīva izmēru
pirmskaitli = pirmskaitlu_masivs(n, drosibas_koeficients)
izvade(pirmskaitli[0])
```

print("Funkcija paredzēja vietas par " + str(pirmskaitli[1]) + " pirmskaitļiem vairāk.")

print(f"{pirmskaitli[1]} liekas nulles masīvā beigās tika nodzestas.")

Testa piemēri:

1)

```
Pimskaitļi līdz 10000:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 16
Funkcija paredzēja vietas par 74 pirmskaitļiem vairāk.
74 liekas nulles masīvā beigās tika nodzestas.
```

2)

```
', 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337,
```

3)

```
, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547,
```

4)

```
49, 9767, 9769, 9781, 9787, 9791, 9803, 9811, 9817, 9829, 9833, 9839, 9851, 9857, 9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941, 9949, 9967, 9973
```

2. uzdevums

Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) saskaitīšanu. Jāveic ievaddatu korektuma pārbaude!

Kods:

Programmas nosaukums: Lielo skaitļu saskaitīšana

2. uzdevums (1MPR08_Vladislavs_Babaņins)

Uzdevuma formulējums: Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) saskaitīšanu.

Jāveic ievaddatu korektuma pārbaude!

Programmas autors: Vladislavs Babaņins

Versija 1.0

import numpy

def is_natural_or_zero_long(s):

```
    # Pārbauda vai simbolu virkne reprezentē naturālo skaitli vai 0, vai nē.
    # Atgriež True, ja virkne reprezentē naturālo skaitli.
    # Atgriež False, ja nereprezentē naturālo skaitli.
    # s - pārbaudama simbolu virkne

    # Noņema no virknes visas sākuma vai beigu atstarpes
    s = s.strip()

    # Pārbauda, vai virkne ir tukša
    if len(s) == 0:
        return False

    # Iet cikliski cauri katrām simbolam simbolu virknē (string'ā)
    for c in s:
        # Ja kāda rakstzīme nav cipars, virkne neatspoguļo naturālu skaitli. return False
        if not c.isdigit():
            return False

    # Virkne atspoguļo naturālu skaitli, ja ietu cauri ciklas netika pamanīts not .isdigit()
    return True


def izvade_bez_komatiem_kopigi(x):
    # Izvada masīva elementus pēc kārtas līdz pedējam kopīgi bez komatiem (izmanto lai izvadītu masīvu kā vienu str skaitli)
    # x - viendimensijas masīvs
    n = len(x)
    s = str(x[0])
    for i in range(1, n):
        s = s + "" + str(x[i])
    print(s[::-1])
```

```
def parveide_sv_to_mas(sv, m):
    # Transformē simbolu virkni par masīvu ar noradīto garumu. Ja garums ir lielāks nekā simbolu virkne, tad beigās tiek pieliktas 0
    # sv - simbolu virkne, kura sastāv no cipariem
    # m - garums, līdz kurām vajag transformēt simbolu virkni sarakstā. Ja garums ir lielāks nekā simbolu virkne, tad beigās tiek pieliktas 0
    n = len(sv)
    a = numpy.arange(m)
    for i in range(n):
        a[i] = int(sv[-i - 1])
    for i in range(n, m):
        a[i] = 0
    return a


def parveide_mas_to_sv(a):
    # Transformē masīvu par simbolu virkni. Ja masīvā priekšā ir 0, tad tas tiek noņemtas
    # a - viendimensijas masīvs
    n = len(a)
    sv = ""
    for i in range(n):
        sv = str(a[i]) + sv

    try:  # try/except gadījumam ja 0 + 0
        while sv[0] == "0":
            sv = sv[1:]
        return sv

    except IndexError:  # Tas ir nepieciešams gadījumam ja lietotājs ievadīs 0 + 0 = 0.
        return "0"  # bez šim rindām būtu kļūda, kad 0 + 0
```

```
def saskaitisana(a, b):
    # Saskaita masīvu a un b izmantojot str. Jāizmanto lielo skaitļu (vismaz ar 50 cipariem) saskaitīšanu
    # a - viendimensijas masīvs
    # b - viendimensijas masīvs
    a = a[::-1]
    b = b[::-1]
    n1 = len(a)
    n2 = len(b)
    if n1 > n2:
        n = n1
    else:
        n = n2
    m1 = parveide_sv_to_mas(a, n)
    m2 = parveide_sv_to_mas(b, n)
    m3 = numpy.zeros(n + 1, dtype=numpy.int_)

    s = 0
    for i in range(n):
        s = s + m1[i] + m2[i]
        m3[i] = s % 10
        s = s // 10
    m3[n] = s

    return parveide_mas_to_sv(m3)


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

```
# ------------------------------------------------------


a = input("Ievadiet 1.saskaitāmo ===> ")
while is_natural_or_zero_long(a) == False:
    a = input("Kļūda! Ievadītam skaitlim jābūt naturālam!\nIevadiet 1.saskaitāmo ===> ")

a = parveide_sv_to_mas(a, len(a))



b = input("Ievadiet 2.saskaitāmo ===> ")
while is_natural_or_zero_long(b) == False:
    b = input("Kļūda! Ievadītam skaitlim jābūt naturālam!\nIevadiet 1.saskaitāmo ===> ")

b = parveide_sv_to_mas(b, len(b))



print("")
izvade_bez_komatiem_kopigi(a)
print("+")
izvade_bez_komatiem_kopigi(b)
print("=")
print(saskaitisana(a, b))
```

Testa piemēri:

1)

```
Ievadiet 1.saskaitāmo ===> 666
Ievadiet 2.saskaitāmo ===> 777

666
+
777
=
1443
```

2)

```
Ievadiet 1.saskaitāmo ===> 1000
Ievadiet 2.saskaitāmo ===> 99

1000
+
99
=
1099
```

3)

```
Ievadiet 1.saskaitāmo ===> 9999
Ievadiet 2.saskaitāmo ===> 9

9999
+
9
=
10008
```

4)

```
Ievadiet 1.saskaitāmo ===> 20
Ievadiet 2.saskaitāmo ===> 1

20
+
1
=
21
```

5)

```
Ievadiet 1.saskaitāmo ===> 123456
Ievadiet 2.saskaitāmo ===> 654321

123456
+
654321
=
777777
```

6)

```
Ievadiet 1.saskaitāmo ===> 999
Ievadiet 2.saskaitāmo ===> 999

999
+
999
=
1998
```

7)

```
Ievadiet 1.saskaitāmo ===> 6666
Ievadiet 2.saskaitāmo ===> 6666

6666
+
6666
=
13332
```

8)

```
Ievadiet 1.saskaitāmo ===> 10
Ievadiet 2.saskaitāmo ===> 0

10
+
0
=
10
```

9)

```
Ievadiet 1.saskaitāmo ===> 5
Ievadiet 2.saskaitāmo ===> 3


5
+
3
=
8
```

10)

```
Ievadiet 1.saskaitāmo izmēru ===> 1
Ievadiet 0.ciparu => 0
Ievadiet 2.saskaitāmo izmēru ===> 1
Ievadiet 0.ciparu => 0

0
+
0
=
0
```

11)

```
Ievadiet 1.saskaitāmo ===> 6999
Ievadiet 2.saskaitāmo ===> 1999999999999999999

6999
+
1999999999999999999
=
2000000000000006998
```

12)

```
Ievadiet 1.saskaitāmo ===> 41646464154163146516534156415634156341635414156456341653415634646345641645646351312312312315416358464631321531635463464121315165345684641352132131531546854413513213521
Ievadiet 2.saskaitāmo ===> 646848564642313154684648413156454198498465165135186496849846513153151351365141654

41646464154163146516534156415634156341635414156456341653415634646345641645646351312312312315416358464631321531635463464121315165345684641352132131531546854413513213521
+
646848564642313154684648413156454198498465165135186496849846513153151351365141654
=
41646464154163146516534156415634156341635414156456341653415634646345641645646351312312959163981000777786006180048619918319813630510819827848981978044700005764878355175
```

13)

```
Ievadiet 1.saskaitāmo ===> 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
Ievadiet 2.saskaitāmo ===> 0

11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
+
0
=
11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

14)

```
Ievadiet 1.saskaitāmo ===> 66666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
Ievadiet 2.saskaitāmo ===> 59999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999

66666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
+
59999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999
=
126666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666665
```

15)

```
Ievadiet 1.saskaitāmo ===> 666
Ievadiet 2.saskaitāmo ===> 66666

666
+
66666
=
67332
```

16)

```
Ievadiet 1.saskaitāmo ===> labi
Kļūda! Ievadītam skaitlim jābūt naturālam!
Ievadiet 1.saskaitāmo ===> 5.5
Kļūda! Ievadītam skaitlim jābūt naturālam!
Ievadiet 1.saskaitāmo ===> -5.5
Kļūda! Ievadītam skaitlim jābūt naturālam!
Ievadiet 1.saskaitāmo ===> -5
Kļūda! Ievadītam skaitlim jābūt naturālam!
Ievadiet 1.saskaitāmo ===> 5
Ievadiet 2.saskaitāmo ===> pieci
Kļūda! Ievadītam skaitlim jābūt naturālam!
Ievadiet 1.saskaitāmo ===> -1
Kļūda! Ievadītam skaitlim jābūt naturālam!
Ievadiet 1.saskaitāmo ===> -2
Kļūda! Ievadītam skaitlim jābūt naturālam!
Ievadiet 1.saskaitāmo ===> 3

5
+
3
=
8
```

3. uzdevums

Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) atņemšanu. Pirms darbības veikšanas jāpārliecinās, ka mazināmai ir lielāks nekā mazinātājs. Jāveic ievaddatu korektuma pārbaude!

Kods:

```
# Programmas nosaukums: Lielo skaitļu atņemšana

# 3. uzdevums (1MPR08_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) atņemšanu.

# Pirms darbības veikšanas jāpārliecinās, ka mazināmai ir lielāks nekā mazinātājs.

# Jāveic ievaddatu korektuma pārbaude!

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import numpy



def izvade_bez_komatiem_kopigi(x):

    # Izvada masīva elementus pēc kārtas līdz pedējam kopīgi bez komatiem (izmanto lai izvadītu masīvu kā vienu str skaitli)

    # x - viendimensijas masīvs

    n = len(x)

    s = str(x[0])

    for i in range(1, n):

        s = s + "" + str(x[i])

    print(s)



def parveide_sv_to_mas(sv, m):

    # Transformē simbolu virkni masīvā ar noradīto garumu. Ja garums ir lielāks nekā simbolu virkne, tad beigās tiek pieliktas 0
```

```
    # sv - simbolu virkne, kura sastāv no cipariem
    # m - garums, līdz kurām vajag transformēt simbolu virkni sarakstā. Ja garums ir lielāks nekā simbolu virkne, tad beigās tiek pieliktas 0
    n = len(sv)
    a = numpy.arange(m)
    for i in range(n):
        a[i] = int(sv[-i - 1])
    for i in range(n, m):
        a[i] = 0
    return a[::-1]


def parveide_mas_to_sv(a):
    # Transformē masīvu simbolu virknē. Ja masīvā priekšā ir 0, tad tas tiek noņemtas
    # a - viendimensijas masīvs
    n = len(a)
    sv = ""
    for i in range(n):
        sv = str(a[i]) + sv
    return sv


def parveide_mas_to_sv_reverse(a):
    # Transformē masīvu simbolu virknē un apgriež to
    # a - viendimensijas masīvs
    n = len(a)
    sv = ""
    for i in range(n):
        sv = str(a[i]) + sv
    return sv[::-1]
```

```
def remove_front_zeros(s):
    # Atgriež simbolu virkni bez nullem priekšā
    # s - simbolu virkne, kura sastav no cipariem, potenciāli ar nullēm priekšā
    i = 0
    while i < len(s) - 1 and s[i] == '0':
        i += 1
    s = s[i:]
    return s



def atnemsana(a, b):
    # Atņem masīvu a no b izmantojot str. Jāizmanto lielo skaitļu (vismaz ar 50 cipariem) atņemšanai. Atgriež simbolu virkni bez nullem priekšā
    # a - viendimensijas masīvs
    # b - viendimensijas masīvs
    if str(a) == str(b):
        return "0"

    n1 = len(a)
    n2 = len(b)
    if n1 > n2:
        n = n1
    else:
        n = n2
    m1 = parveide_sv_to_mas(a, n)
    m2 = parveide_sv_to_mas(b, n)
    m3 = numpy.zeros(n, dtype=numpy.int_)

    for i in range(-1, -n1, -1):
        if m1[i] < m2[i]:
```

```
            m1[i - 1] = m1[i - 1] - 1
            m1[i] = m1[i] + 10

    for i in range(n):
        m3[i] = m1[i] - m2[i]

    a = parveide_mas_to_sv(m3)
    a = a[::-1]
    a = remove_front_zeros(a)
    return a



def is_natural_or_zero_long(s):
    # Pārbauda vai simbolu virkne reprezentē naturālo skaitli vai 0, vai nē.
    # Atgriež True, ja virkne reprezentē naturālo skaitli.
    # Atgriež False, ja nereprezentē naturālo skaitli.
    # s - pārbaudāma simbolu virkne

    # Noņema no virknes visas sākuma vai beigu atstarpes
    s = s.strip()

    # Pārbauda, vai virkne ir tukša
    if len(s) == 0:
        return False

    # Iet cikliski cauri katrām simbolam simbolu virknē (string'ā)
    for c in s:
        # Ja kāda rakstzīme nav cipars, virkne neatspoguļo naturālu skaitli. return False
        if not c.isdigit():
            return False
```

```
        # Virkne atspoguļo naturālu skaitli, ja ietu cauri ciklas netika pamanīts not .isdigit()
        return True


    def is_a_bigger_b(a, b):
        # Vai masīvs a ir lielāks vai vienāds nekā masīvs b, ja jā, tad return True. Ja nē, tad return False
        # a - pirmais viendimensijas masīvs
        # b - otrais viendimensijas masīvs
        a = parveide_mas_to_sv_reverse(a)
        b = parveide_mas_to_sv_reverse(b)
        if int(a) < int(b):
            return False
        else:
            return True


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


    a = input("Ievadiet mazināmo ===> ")
    while is_natural_or_zero_long(a) == False:
        a = input("Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!\nIevadiet mazināmo ===> ")

    a = parveide_sv_to_mas(a, len(a))


    b = input("Ievadiet mazinātāju ===> ")
    while is_natural_or_zero_long(b) == False:
```

```
    b = input("Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!\nIevadiet mazinātāju
===> ")


    b = parveide_sv_to_mas(b, len(b))



    if is_a_bigger_b(a, b) == False:
        print("Kļūda! Mazināmam jābūt lielākam par mazinātāju!")
    else:
        print("")
        izvade_bez_komatiem_kopigi(a)
        print("-")
        izvade_bez_komatiem_kopigi(b)
        print("=")
        print(atnemsana(a, b))
```

Testa piemēri:

1)

```
Ievadiet mazināmo ===> 100
Ievadiet mazinātāju ===> 50

100
-
50
=
50
```

2)

```
Ievadiet mazināmo ===> 100
Ievadiet mazinātāju ===> 100

100
-
100
=
0
```

3)

```
Ievadiet mazināmo ===> 999
Ievadiet mazinātāju ===> 678

999
-
678
=
321
```

4)

```
Ievadiet mazināmo ===> 955
Ievadiet mazinātāju ===> 788

955
-
788
=
167
```

5)

```
Ievadiet mazināmo ===> 666
Ievadiet mazinātāju ===> 599

666
-
599
=
67
```

6)

```
Ievadiet mazināmo ===> 600
Ievadiet mazinātāju ===> 700
Kļūda! Mazināmam jābūt lielākam par mazinātāju!
```

7)

```
Ievadiet mazināmo ===> pieci
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> -5
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> 5
Ievadiet mazinātāju ===> 5

5
-
5
=
0
```

8)

```
Ievadiet mazināmo ===> 7777
Ievadiet mazinātāju ===> 6999

7777
-
6999
=
778
```

9)

```
Ievadiet mazināmo ===> 55555555
Ievadiet mazinātāju ===> 66666


55555555
-
66666
=
55488889
```

10)

```
Ievadiet mazināmo ===> 20
Ievadiet mazinātāju ===> 0

20
-
0
=
20
```

11)

```
Ievadiet mazināmo ===> 123
Ievadiet mazinātāju ===> 123

123
-
123
=
0
```

12)

```
Ievadiet mazināmo ===> 1
Ievadiet mazinātāju ===> 1


1
-
1
=
0
```

13)

```
Ievadiet mazināmo ===> 0
Ievadiet mazinātāju ===> 0

0
-
0
=
0
```

14)

```
Ievadiet mazināmo ===> 555555
Ievadiet mazinātāju ===> 6000000
Kļūda! Mazināmam jābūt lielākam par mazinātāju!
```

15)

```
Ievadiet mazināmo ===> pieci
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> 45.24
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> 23...32
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> .
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> labi
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> -2
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazināmo ===> 10
Ievadiet mazinātāju ===> -2
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazinātāju ===> 12.2
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazinātāju ===> pieci
Kļūda! Skaitļa izmēram jābūt naturālam skaitlim vai nulle!
Ievadiet mazinātāju ===> 5

10
-
5
=
5
```

4. uzdevums

Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) reizināšanu, izmantojot naivo skolas reizināšanas algoritmu. Jāveic ievaddatu korektuma pārbaude!

Kods:

```
# Programmas nosaukums: Lielo skaitļu reizināšana

# 4. uzdevums (1MPR08_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) reizināšanu,

# izmantojot naivo skolas reizināšanas algoritmu.

# Jāveic ievaddatu korektuma pārbaude!

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import numpy



def is_natural_or_zero_long(s):

    # Pārbauda vai simbolu virkne reprezentē naturālo skaitli vai 0, vai nē.

    # Atgriež True, ja virkne reprezentē naturālo skaitli.

    # Atgriež False, ja nereprezentē naturālo skaitli.

    # s - pārbaudama simbolu virkne


    # Noņema no virknes visas sākuma vai beigu atstarpes

    s = s.strip()


    # Pārbauda, vai virkne ir tukša

    if len(s) == 0:

        return False


    # Iet cikliski cauri katrām simbolam simbolu virknē (string'ā)
```

```
    for c in s:
        # Ja kāda rakstzīme nav cipars, virkne neatspoguļo naturālu skaitli. return False
        if not c.isdigit():
            return False

    # Virkne atspoguļo naturālu skaitli, ja ietu cauri ciklas netika pamanīts not .isdigit()
    return True


def izvade_bez_komatiem_kopigi(x):
    # Izvada masīva elementus pēc kārtas līdz pedējam kopīgi bez komatiem (izmanto, lai izvadītu masīvu kā vienu str skaitli)
    # x - viendimensijas masīvs
    n = len(x)
    s = str(x[0])
    for i in range(1, n):
        s = s + "" + str(x[i])

    s = s[::-1]
    print(s)


def parveide_sv_to_mas(sv, m):
    # Transformē simbolu virkni masīvā ar noradīto garumu. Ja garums ir lielāks nekā simbolu virkne, tad beigās tiek pieliktas 0
    # sv - simbolu virkne, kura sastāv no cipariem
    # m - (vēlamais masīva garums) garums, līdz kurām vajag transformēt simbolu virkni sarakstā. Ja garums ir lielāks nekā simbolu virkne, tad beigās tiek pieliktas 0
    n = len(sv)
    a = numpy.arange(m)
    for i in range(n):
        a[i] = int(sv[-i - 1])
```

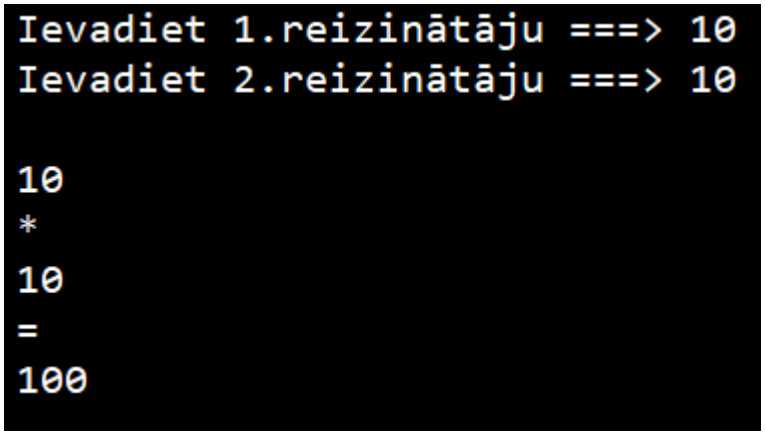
```
    for i in range(n, m):
        a[i] = 0
    return a


def parveide_mas_to_sv(a):
    # Transformē masīvu simbolu virknē pretējā secībā. Ja masīvā priekšā ir 0, tad tas tiek noņemtas
    # a - viendimensijas masīvs
    n = len(a)
    sv = ""
    for i in range(n):
        sv = str(a[i]) + sv

    try:  # try/except gadījumam ja a * 0, a - jebkurš skaitlis
        while sv[0] == "0":
            sv = sv[1:]
        return sv

    except IndexError:  # Tas ir nepieciešams gadījumam ja lietotājs ievadīs a *  0 = 0, a - jebkurš
skaitlis
        return "0"  # bez šim rindām būtu kļūda, kad a * 0, a - jebkurš skaitlis


def reizinasana(a, b):
    # Sareizina masīvu a ar b izmantojot str. Jāizmanto lielo skaitļu (vismaz ar 50 cipariem) reizināšanai
    # a - viendimensijas masīvs
    # b - viendimensijas masīvs

    a = a[::-1]
    b = b[::-1]
    n1 = len(a)
```

```
    n2 = len(b)
    if n1 > n2:
        n = n1
    else:
        n = n2

    m1 = parveide_sv_to_mas(a, n)
    m2 = parveide_sv_to_mas(b, n)
    m3 = numpy.zeros(2 * n, dtype=numpy.int_)
    s = 0
    for j in range(2 * n - 1):
        for i in range(n):
            if (0 <= j - i) and (j - i <= n - 1):
                s = s + m1[i] * m2[j - i]
        m3[j] = s % 10
        s = s // 10
    m3[n * 2 - 1] = s

    return parveide_mas_to_sv(m3)


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


a = input("Ievadiet 1.reizinātāju ===> ")
while is_natural_or_zero_long(a) == False:
    a = input("Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.\nIevadiet 1.reizinātāju ===> ")

a = parveide_sv_to_mas(a, len(a))
```

```
b = input("Ievadiet 2.reizinātāju ===> ")
while is_natural_or_zero_long(b) == False:
    b = input("Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.\nIevadiet 1.reizinātāju ===> ")

b = parveide_sv_to_mas(b, len(b))

print("")
izvade_bez_komatiem_kopigi(a)
print("*")
izvade_bez_komatiem_kopigi(b)
print("=")
print(reizinasana(a, b))
```

Testa piemēri:

1)



2)

```
Ievadiet 1.reizinātāju ===> 333
Ievadiet 2.reizinātāju ===> 66

333
*
66
=
21978
```

3)

```
Ievadiet 1.reizinātāju ===> 0
Ievadiet 2.reizinātāju ===> 0

0
*
0
=
0
```

4)

```
Ievadiet 1.reizinātāju ===> 666
Ievadiet 2.reizinātāju ===> 0


666
*
0
=
0
```

5)

```
Ievadiet 1.reizinātāju ===> 123456
Ievadiet 2.reizinātāju ===> 65431

123456
*
65431
=
8077849536
```

6)

```
Ievadiet 1.reizinātāju ===> 10
Ievadiet 2.reizinātāju ===> 99

10
*
99
=
990
```

7)

```
Ievadiet 1.reizinātāju ===> 456
Ievadiet 2.reizinātāju ===> 123425669871

456
*
123425669871
=
56282105461176
```

8)

```
Ievadiet 1.reizinātāju ===> 777777
Ievadiet 2.reizinātāju ===> 1

777777
*
1
=
777777
```

9)

```
Ievadiet 1.reizinātāju ===> 658
Ievadiet 2.reizinātāju ===> 987

658
*
987
=
649446
```

10)

```
Ievadiet 1.reizinātāju ===> 555
Ievadiet 2.reizinātāju ===> 555

555
*
555
=
308025
```

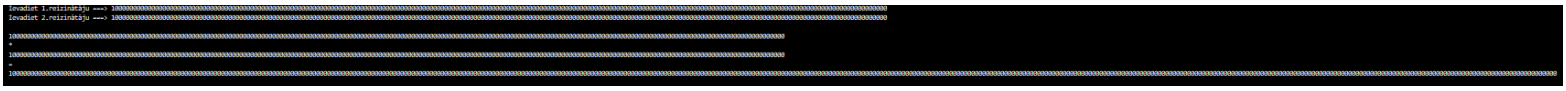
11)

```
Ievadiet 1.reizinātāju ===> 555
Ievadiet 2.reizinātāju ===> 5555

555
*
5555
=
3083025
```

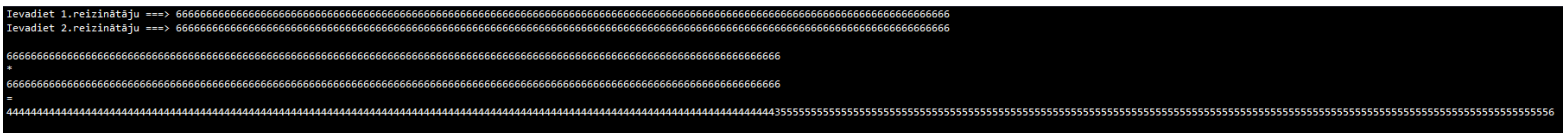
12)

```
Ievadiet 1.reizinātāju ===> 123456789
Ievadiet 2.reizinātāju ===> 123456789

123456789
*
123456789
=
15241578750190521
```

13)



14)



15)

```
Ievadiet 1.reizinātāju ===> 999999999999999999999999
Ievadiet 2.reizinātāju ===> 555555555555555555555555

999999999999999999999999
*
555555555555555555555555
=
555555555555555555555549994444444444444444444445
```

16)

```
Ievadiet 1.reizinātāju ===> 77777777777777777777777777
Ievadiet 2.reizinātāju ===> 0

77777777777777777777777777
*
0
=
0
```

17)

```
Ievadiet 1.reizinātāju ===> 666
Ievadiet 2.reizinātāju ===> 666

666
*
666
=
443556
```

18)

```
Ievadiet 1.reizinātāju ===> 1111
Ievadiet 2.reizinātāju ===> 1111

1111
*
1111
=
1234321
```

19)

```
Ievadiet 1.reizinātāju ===> -2
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> 12.2
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> -2
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> pieci
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> .
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===>
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> 5
Ievadiet 2.reizinātāju ===> pieci
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> labi
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> -2
Kļūda! Ievadītam skaitlim jābūt naturālam vai 0.
Ievadiet 1.reizinātāju ===> 0

5
*
0
=
0
```

5. uzdevums

Sastādīt programmu, kas nodrošina divdimensiju matricas elementu ievadi, kas ir veseli skaitļi. Matricas rindu un kolonnu skaitu ievada lietotājs. Pēc matricas elementu ievades tiek nodrošināta matricas elementu izvade tabulas veidā, pieņemot, ka neviens no matricas elementiem nesastāv no ne vairāk kā 4 cipariem, kā arī lielāko un mazāko matricas elementa vērtību un tā atrašanās vietu, noradot atbilstošo rindu un kolonnu.

Izvades piemērs (* - tukšums):

****1**137***23*8765

*1234***53****1**876

13*2***34****5

Mazākais elements ir 1, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.

Lielākais elements ir 8765, un tas atrodas 1.rindas un 4.kolonnas krustpunktā.

Kods:

```
# Programmas nosaukums: Matricas max un min elements

# 5. uzdevums (1MPR08_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas nodrošina divdimensiju matricas elementu ievadi,

# kas ir veseli skaitļi. Matricas rindu un kolonnu skaitu ievada lietotājs.

# Pēc matricas elementu ievades tiek nodrošināta matricas elementu izvade_matrica_int tabulas veidā,

# pieņemot, ka neviens no matricas elementiem nesastāv no ne vairāk kā 4 cipariem, kā arī

# lielāko un mazāko matricas elementa vērtību un tā atrašanās vietu, noradot atbilstošo rindu un kolonnu.

# 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 is_whole(n):
    # Pārbauda vai simbolu virkne ir vesels skaitlis vai nav
    # Ja ir vesels skaitlis, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    try:
        n = int(n)
    except:
        return False
    else:
        return True


def ievade_matrica(n, m):
    # Lietotājs var ievādīt nXm matricas elementus un funkcija atgriež divdimensijas masīvu ar n rindam un m kolonnam ar ievadītām vērtībam
    # n - naturāls skaitlis, kurš nosaka matricas rindas skaitu
    # m - naturāls skaitlis, kurš nosaka matricas kolonnas skaitu
    a = numpy.empty((n, m), dtype=int)
    for i in range(n):
        for j in range(m):
            temp = input("Ievadiet matricas elememtu a(" + str(i) + "," + str(j) + ") ===> ")
            while is_whole(temp) == False:
                temp = input("Kļūda! Ievadītais elements nav vesels skaitlis!\nIevadiet matricas elememtu a(" + str(i) + "," + str(j) + ") ===> ")
            a[i, j] = int(temp)
```

```
    return a


def izvade_matrica_int(a):
    # Atgriež divdimensiju masīvu (matricu), tabulas veidā, str formāta, kur katra rinda ir atdalīta ar jauno rindkopu
    # a - divdimensijas masīvs (matrica)
    n = a.shape[0]  # "x axis" masīva izmēri
    m = a.shape[1]  # "y axis" masīva izmēri

    s = ""
    for i in range(n):
        for j in range(m):
            s = s + "{:8.2f}".format(int(a[i, j]))
        s = s + "\n"
    return s


def matricas_izvade_4_cipari(m):
    # Izvada divdimensijas masīvu tabulas veidā tā, ka neviens no matricas elementiem nesastāv no ne vairāk kā 4 cipariem
    # Tiek nodrošināta matricas elementu izvade tabulas veidā, pieņemot, ka neviens no matricas elementiem nesastāv no ne vairāk kā 4 cipariem
    # Atgriež matricas elementu izvade tabulas veidā
    # m - divdimensijas masīvs
    nm = numpy.shape(m)
    sv = ""
    for x in range(nm[0]):
        for y in range(nm[1]):
            for i in range(5 - len(str(m[x, y]))):
                sv = sv + " "
            sv = sv + str(m[x, y])
```

```
        sv = sv + "\n"
    nn = len(sv)
    return sv[:nn - 1]


def max_value_in_2_dimensional_array_and_its_coords(a):
    # Atgriež tuple (max_value, coords) ar divdimensijas masīva (matricas) maksimālo elementu vērtību un tas koordinātu [i,j] pēc rindām un kolonnam
    # Atgriež maksimālo vērtību un kur tā vērtība atrodas pēc koordinātam
    # a - divdimensijas masīvs
    max_value = a[0][0]
    coords = [0, 0]

    for i in range(len(a)):
        for j in range(len(a[0])):
            if a[i][j] > max_value:
                max_value = a[i][j]
                coords = [i, j]

    return (max_value, coords)


def min_value_in_2_dimensional_array_and_its_coords(a):
    # Atgriež kortežu (tuple) (min_value, coords) ar divdimensijas masīva (matricas) minimālo elementu vērtību un tas koordinātu [i,j] pēc rindām un kolonnam
    # Atgriež minimālo vērtību un kur tā vērtība atrodas pēc koordinātam
    # a - divdimensijas masīvs
    min_value = a[0][0]
    coords = [0, 0]
    for i in range(len(a)):
        for j in range(len(a[0])):
            if a[i][j] < min_value:
```

```
                min_value = a[i][j]
                coords = [i, j]

        return (min_value, coords)



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



n = input("Ievadiet matricas rindu skaitu ===> ")
while is_natural(n) == False:
    n = input("Kļūda! Matricas rindu skaitam jābūt naturālam skaitlim!\nIevadiet matricas rindu skaitu ===> ")
n = int(n)



m = input("Ievadiet matricas kolonnu skaitu ===> ")
while is_natural(m) == False:
    m = input("Kļūda! Matricas kolonnu skaitam jābūt naturālam skaitlim!\nIevadiet matricas kolonnu skaitu ===> ")
m = int(m)

a = ievade_matrica(n, m)

print(matricas_izvade_4_cipari(a))

max_value_and_coord = max_value_in_2_dimensional_array_and_its_coords(a)
min_value_and_coord = min_value_in_2_dimensional_array_and_its_coords(a)
```

print("Mazākais elements ir " + str(min_value_and_coord[0]) + ", un tas atrodas " + str(min_value_and_coord[1][0] + 1) + ".rindas un " + str(min_value_and_coord[1][1] + 1) + ".kolonnas krustpunktā.")

print("Lielākais elements ir " + str(max_value_and_coord[0]) + ", un tas atrodas " + str(max_value_and_coord[1][0] + 1) + ".rindas un " + str(max_value_and_coord[1][1] + 1) + ".kolonnas krustpunktā.")

Testa piemēri:

1)

```
Ievadiet matricas rindu skaitu ===> 3
Ievadiet matricas kolonnu skaitu ===> 4
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 137
Ievadiet matricas elememtu a(0,2) ===> 23
Ievadiet matricas elememtu a(0,3) ===> 8765
Ievadiet matricas elememtu a(1,0) ===> 1234
Ievadiet matricas elememtu a(1,1) ===> 53
Ievadiet matricas elememtu a(1,2) ===> 1
Ievadiet matricas elememtu a(1,3) ===> 876
Ievadiet matricas elememtu a(2,0) ===> 13
Ievadiet matricas elememtu a(2,1) ===> 2
Ievadiet matricas elememtu a(2,2) ===> 34
Ievadiet matricas elememtu a(2,3) ===> 5
    1  137   23 8765
 1234   53    1  876
   13    2   34    5
Mazākais elements ir 1, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 8765, un tas atrodas 1.rindas un 4.kolonnas krustpunktā.
```

2)

```
Ievadiet matricas rindu skaitu ===> 2
Ievadiet matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(1,0) ===> 3
Ievadiet matricas elememtu a(1,1) ===> 4
    1    2
    3    4
Mazākais elements ir 1, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 4, un tas atrodas 2.rindas un 2.kolonnas krustpunktā.
```

3)

```
Ievadiet matricas rindu skaitu ===> 2
Ievadiet matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 2
Ievadiet matricas elememtu a(0,1) ===> 6
Ievadiet matricas elememtu a(0,2) ===> 4
Ievadiet matricas elememtu a(1,0) ===> 1
Ievadiet matricas elememtu a(1,1) ===> 2
Ievadiet matricas elememtu a(1,2) ===> 0
    2    6    4
    1    2    0
Mazākais elements ir 0, un tas atrodas 2.rindas un 3.kolonnas krustpunktā.
Lielākais elements ir 6, un tas atrodas 1.rindas un 2.kolonnas krustpunktā.
```

4)

```
Ievadiet matricas rindu skaitu ===> 2
Ievadiet matricas kolonnu skaitu ===> 4
Ievadiet matricas elememtu a(0,0) ===> 2
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 2
Ievadiet matricas elememtu a(0,3) ===> 2
Ievadiet matricas elememtu a(1,0) ===> 2
Ievadiet matricas elememtu a(1,1) ===> 2
Ievadiet matricas elememtu a(1,2) ===> 2
Ievadiet matricas elememtu a(1,3) ===> 2
    2    2    2    2
    2    2    2    2
Mazākais elements ir 2, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 2, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
```

5)

```
Ievadiet matricas rindu skaitu ===> 6
Ievadiet matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(1,0) ===> 2
Ievadiet matricas elememtu a(2,0) ===> 55
Ievadiet matricas elememtu a(3,0) ===> -55
Ievadiet matricas elememtu a(4,0) ===> 1
Ievadiet matricas elememtu a(5,0) ===> 1
    1
    2
   55
  -55
    1
    1
Mazākais elements ir -55, un tas atrodas 4.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 55, un tas atrodas 3.rindas un 1.kolonnas krustpunktā.
```

6)

```
Ievadiet matricas rindu skaitu ===> -1
Kļūda! Matricas rindu skaitam jābūt naturālam skaitlim!
Ievadiet matricas rindu skaitu ===> pieci
Kļūda! Matricas rindu skaitam jābūt naturālam skaitlim!
Ievadiet matricas rindu skaitu ===> 1
Ievadiet matricas kolonnu skaitu ===> -12
Kļūda! Matricas kolonnu skaitam jābūt naturālam skaitlim!
Ievadiet matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> -1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 12.2
Kļūda! Ievadītais elements nav vesels skaitlis!
Ievadiet matricas elememtu a(0,2) ===> 12
   -1    2   12
Mazākais elements ir -1, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 12, un tas atrodas 1.rindas un 3.kolonnas krustpunktā.
```

7)

```
Ievadiet matricas rindu skaitu ===> 0
Kļūda! Matricas rindu skaitam jābūt naturālam skaitlim!
Ievadiet matricas rindu skaitu ===> viens
Kļūda! Matricas rindu skaitam jābūt naturālam skaitlim!
Ievadiet matricas rindu skaitu ===> -1
Kļūda! Matricas rindu skaitam jābūt naturālam skaitlim!
Ievadiet matricas rindu skaitu ===> 1
Ievadiet matricas kolonnu skaitu ===> 1.2
Kļūda! Matricas kolonnu skaitam jābūt naturālam skaitlim!
Ievadiet matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 9
    9
Mazākais elements ir 9, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 9, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
```

8)

```
Ievadiet matricas rindu skaitu ===> 1
Ievadiet matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
    1    2
Mazākais elements ir 1, un tas atrodas 1.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 2, un tas atrodas 1.rindas un 2.kolonnas krustpunktā.
```

9)

```
Ievadiet matricas rindu skaitu ===> 5
Ievadiet matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 12
Ievadiet matricas elememtu a(1,0) ===> -12
Ievadiet matricas elememtu a(1,1) ===> 5
Ievadiet matricas elememtu a(2,0) ===> 6
Ievadiet matricas elememtu a(2,1) ===> 8
Ievadiet matricas elememtu a(3,0) ===> 2
Ievadiet matricas elememtu a(3,1) ===> 0
Ievadiet matricas elememtu a(4,0) ===> 0
Ievadiet matricas elememtu a(4,1) ===> 0
    1   12
  -12    5
    6    8
    2    0
    0    0
Mazākais elements ir -12, un tas atrodas 2.rindas un 1.kolonnas krustpunktā.
Lielākais elements ir 12, un tas atrodas 1.rindas un 2.kolonnas krustpunktā.
```

10)

```
Ievadiet matricas rindu skaitu ===> 3
Ievadiet matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 2
Ievadiet matricas elememtu a(0,1) ===> 5
Ievadiet matricas elememtu a(0,2) ===> 6
Ievadiet matricas elememtu a(1,0) ===> 8
Ievadiet matricas elememtu a(1,1) ===> 9
Ievadiet matricas elememtu a(1,2) ===> 2
Ievadiet matricas elememtu a(2,0) ===> 0
Ievadiet matricas elememtu a(2,1) ===> 0
Ievadiet matricas elememtu a(2,2) ===> -12
    2    5    6
    8    9    2
    0    0  -12
Mazākais elements ir -12, un tas atrodas 3.rindas un 3.kolonnas krustpunktā.
Lielākais elements ir 9, un tas atrodas 2.rindas un 2.kolonnas krustpunktā.
```

6. uzdevums

Sastādīt programmu, kas nodrošina divu matricu elementu ievadi, kuri ir veseli skaitļi (matricu izmēri tiek izvēlēti tā), lai varētu veikt kādu no zemāk minētajām darbībām.

1. – sareizina abu matricu atbilstošās elementu vērtības.
2. – veic divu matricu reizināšanu.

Programmu var veidot tā, ka katrs no punktiem tiek pildīts atsevišķi vai arī vispirms ievada, kuru darbību veiks un tad izpilda tikai izvēlēto darbību.

Kods:

```
# Programmas nosaukums: Matricas reizinājums un matricu atbilstošo elementu reizinājums

# 6. uzdevums (1MPR08_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas nodrošina divdimensiju matricas elementu ievadi,

# kas ir veseli skaitļi. Matricas rindu un kolonnu skaitu ievada lietotājs.

# Pēc matricas elementu ievades tiek nodrošināta matricas elementu izvade_matrica_int tabulas veidā,

# pieņemot, ka neviens no matricas elementiem nesastāv no ne vairāk kā 4 cipariem, kā arī

# lielāko un mazāko matricas elementa vērtību un tā atrašanās vietu, noradot atbilstošo rindu un kolonnu.

# 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 is_whole(n):
        # Pārbauda vai simbolu virkne ir vesels skaitlis vai nav
        # Ja ir vesels skaitlis, tad True. Ja nav tad False.
        # n - simbolu virkne, kuru pārbauda.
        try:
            n = int(n)
        except:
            return False
        else:
            return True


    def ievade_matrica(n, m):
        # Lietotājs var ievādīt nXm matricas elementus un funkcija atgriež divdimensijas masīvu ar n rindam un m kolonnam ar ievadītām vērtībam
        # n - naturāls skaitlis, kurš nosaka matricas rindas skaitu
        # m - naturāls skaitlis, kurš nosaka matricas kolonnas skaitu
        a = numpy.empty((n, m), dtype=int)
        for i in range(n):
            for j in range(m):
                temp = input("Ievadiet matricas elememtu a(" + str(i) + "," + str(j) + ") ===> ")
                while is_whole(temp) == False:
                    temp = input("Kļūda! Ievadītais elements nav vesels skaitlis!\nIevadiet matricas elememtu a(" + str(i) + "," + str(j) + ") ===> ")
                a[i, j] = int(temp)
        return a


    def izvade_matrica_int(a):
```

```
    # Atgriež divdimensiju masīvu (matricu), tabulas veidā, str formāta, kur katra rinda ir atdalīta ar jauno rindkopu
    # a - divdimensijas masīvs (matrica)
    n = a.shape[0]  # "x axis" masīva izmēri
    m = a.shape[1]  # "y axis" masīva izmēri

    s = ""
    for i in range(n):
        for j in range(m):
            s = s + "{:8.2f}".format(int(a[i, j]))
            # s = s + "   " + str(int(a[i,j])) #"{:8.2f}".format(a[i,j], dtype=int)
        s = s + "\n"
    return s


def matricas_izvade_4_cipari(m):
    # Izvada divdimensijas masīvu tabulas veidā tā, ka neviens no matricas elementiem nesastāv no ne vairāk kā 4 cipariem
    # Tiek nodrošināta matricas elementu izvade_matrica_int tabulas veidā, pieņemot, ka neviens no matricas elementiem nesastāv no ne vairāk kā 4 cipariem
    # Atgriež matricas elementu izvade_matrica_int tabulas veidā
    # m - divdimensijas masīvs
    nm = (numpy.shape(m))
    sv = ""
    for x in range(nm[0]):
        for y in range(nm[1]):
            for i in range(5 - len(str(m[x, y]))):
                sv = sv + " "
            sv = sv + str(m[x, y])
        sv = sv + "\n"
    nn = len(sv)
    return sv[:nn - 1]
```

```
def check_fake_matrix_multiplication(n1, m1, n2, m2):
    # Pārbauda vai divas matricas (divdimensiju masīvi) ir ar vienādu izmēru. Pārbauda vai sakrīt 1.matricas rindas skaits ar 2.matricas rindas skaitu.
    # un pārbauda vai 1.matricas kolonnu skaits sakrīt ar 2.matricas kolonnu skaitu. Ja abas prasības izpildās, tad return True. Ja kaut viena neizpildās, tad return False.
    # Tiek izmantota, lai pārbaudītu vai var sareizināt matricas atbilstošus elementus vai nē.
    # n1 - 1.matricas rindu skaits
    # m1 - 1.matricas kolonnu skaits
    # n2 - 2.matricas rindu skaits
    # m2 - 2.matricas kolonnu skaits
    if n1 == n2 and m1 == m2:
        return True
    else:
        return False


def check_matrix_multiplication(n1, m1, n2, m2):
    # Pārbauda vai 1.matricas kolonnu skaits sakrīt ar 2.matricas rindu skaitu. Ja sakrīt, tad return True. Ja nē, tad return False.
    # Tiek izmantots, lai pārbaudītu vai ir iespējams sareizināt divas matricas.
    # n1 - 1.matricas rindu skaits (tā ne uz ko neietekme)
    # m1 - 1.matricas kolonnu skaits
    # n2 - 2.matricas rindu skaits
    # m2 - 2.matricas kolonnu skaits (tā ne uz ko neietekme)
    if m1 == n2:
        return True
    else:
        return False
```

```
def matrix_multiplication_integer(a, b):
    # Sareizina divu divdimensijas masīvus (divas matricas) a ar b.
    # Atgriež divdimensiju masīvu, vai ja nevar sareizināt atgriež "Kļūda"
    # a - divdimensijas masīvs
    # b - divdimensijas masīvs
    n1 = a.shape[0]
    m1 = a.shape[1]
    n2 = b.shape[0]
    m2 = b.shape[1]
    if m1 == n2:
        c = numpy.zeros((n1, m2), numpy.int_)
        for i in range(n1):
            for j in range(m2):
                for k in range(m1):
                    c[i, j] = c[i, j] + a[i, k] * b[k, j]
        return c
    else:
        return "Kļūda"


def fake_matrix_multiplication(a, b):
    # Sareizina divas matricas atbilstošus elementus un atgriež atbilstošu divdimensiju masīvu
    # a - divdimensiju masīvs
    # b - divdimensiju masīvs
    c = numpy.empty((n1, m1), dtype=int)

    for i in range(n1):
        for j in range(m2):
            c[i, j] = a[i][j] * b[i][j]

    return c
```

```
# ------------------------------------------------------------

# Galvenā programmas daļa

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



n1 = input("Ievadiet 1.matricas rindu skaitu ===> ")

while is_natural(n1) == False:

    n1 = input("Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!\nIevadiet 1.matricas rindu skaitu ===> ")

n1 = int(n1)



m1 = input("Ievadiet 1.matricas kolonnu skaitu ===> ")

while is_natural(m1) == False:

    m1 = input("Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!\nIevadiet 1.matricas kolonnu skaitu ===> ")

m1 = int(m1)



a = ievade_matrica(n1, m1)



n2 = input("\nIevadiet 2.matricas rindu skaitu ===> ")

while is_natural(n2) == False:

    n2 = input("Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!\nIevadiet 2.matricas rindu skaitu ===> ")

n2 = int(n2)



m2 = input("Ievadiet 2.matricas kolonnu skaitu ===> ")

while is_natural(m2) == False:

    m2 = input("Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!\nIevadiet 2.matricas kolonnu skaitu ===> ")
```

```
    m2 = int(m2)

    b = ievade_matrica(n2, m2)

    print("\nPirmā ievadīta matrica:")
    print(matricas_izvade_4_cipari(a))

    print("\nOtrā ievadīta matrica:")
    print(matricas_izvade_4_cipari(b))

    if check_fake_matrix_multiplication(n1, m1, n2, m2):
        print("\nAbu matricu atbilstošās elementu reizinājums:")
        c = fake_matrix_multiplication(a, b)
        print(matricas_izvade_4_cipari(c))
    else:
        print("\nKļūda! Šādas matricas nevar sareizināt atbilstošus elementus.")

    if check_matrix_multiplication(n1, m1, n2, m2):
        print("\nDivu matricu reizinājums:")
        d = matrix_multiplication_integer(a, b)
        print(matricas_izvade_4_cipari(d))
    else:
        print("\nKļūda! Šādas matricas nevar sareizināt, jo 1.matricas kolonnu skaits nav vienāds ar
2.matricas rindas skaitu.")
```

Testa piemēri:

1)

```
Ievadiet 1.matricas rindu skaitu ===> 2
Ievadiet 1.matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(1,0) ===> 3
Ievadiet matricas elememtu a(1,1) ===> 4

Ievadiet 2.matricas rindu skaitu ===> 2
Ievadiet 2.matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(1,0) ===> 3
Ievadiet matricas elememtu a(1,1) ===> 4

Pirmā ievadīta matrica:
    1    2
    3    4

Otrā ievadīta matrica:
    1    2
    3    4

Abu matricu atbilstošās elementu reizinājums:
    1    4
    9   16

Divu matricu reizinājums:
    7   10
   15   22
```

2)

```
Ievadiet 1.matricas rindu skaitu ===> 1
Ievadiet 1.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 3

Ievadiet 2.matricas rindu skaitu ===> 3
Ievadiet 2.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(1,0) ===> 2
Ievadiet matricas elememtu a(2,0) ===> 3

Pirmā ievadīta matrica:
    1    2    3

Otrā ievadīta matrica:
    1
    2
    3

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Divu matricu reizinājums:
   14
```

3)

```
Ievadiet 1.matricas rindu skaitu ===> 3
Ievadiet 1.matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 2
Ievadiet matricas elememtu a(0,1) ===> 5
Ievadiet matricas elememtu a(1,0) ===> 6
Ievadiet matricas elememtu a(1,1) ===> 8
Ievadiet matricas elememtu a(2,0) ===> 1
Ievadiet matricas elememtu a(2,1) ===> 3

Ievadiet 2.matricas rindu skaitu ===> 2
Ievadiet 2.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 3
Ievadiet matricas elememtu a(1,0) ===> 4
Ievadiet matricas elememtu a(1,1) ===> 5
Ievadiet matricas elememtu a(1,2) ===> 6

Pirmā ievadīta matrica:
    2    5
    6    8
    1    3

Otrā ievadīta matrica:
    1    2    3
    4    5    6

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Divu matricu reizinājums:
   22   29   36
   38   52   66
   13   17   21
```

4)

```
Ievadiet 1.matricas rindu skaitu ===> 3
Ievadiet 1.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 0
Ievadiet matricas elememtu a(0,2) ===> 0
Ievadiet matricas elememtu a(1,0) ===> 0
Ievadiet matricas elememtu a(1,1) ===> 1
Ievadiet matricas elememtu a(1,2) ===> 0
Ievadiet matricas elememtu a(2,0) ===> 0
Ievadiet matricas elememtu a(2,1) ===> 0
Ievadiet matricas elememtu a(2,2) ===> 1

Ievadiet 2.matricas rindu skaitu ===> 2
Ievadiet 2.matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(1,0) ===> 3
Ievadiet matricas elememtu a(1,1) ===> 4

Pirmā ievadīta matrica:
    1    0    0
    0    1    0
    0    0    1

Otrā ievadīta matrica:
    1    2
    3    4

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Kļūda! Šādas matricas nevar sareizināt, jo 1.matricas kolonnu skaits nav vienāds ar 2.matricas rindas skaitu.
```

5)

```
Ievadiet 1.matricas rindu skaitu ===> 3
Ievadiet 1.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 3
Ievadiet matricas elememtu a(1,0) ===> 4
Ievadiet matricas elememtu a(1,1) ===> 5
Ievadiet matricas elememtu a(1,2) ===> 6
Ievadiet matricas elememtu a(2,0) ===> 7
Ievadiet matricas elememtu a(2,1) ===> 0
Ievadiet matricas elememtu a(2,2) ===> 0

Ievadiet 2.matricas rindu skaitu ===> 3
Ievadiet 2.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 6
Ievadiet matricas elememtu a(0,1) ===> 6
Ievadiet matricas elememtu a(0,2) ===> 6
Ievadiet matricas elememtu a(1,0) ===> 6
Ievadiet matricas elememtu a(1,1) ===> 6
Ievadiet matricas elememtu a(1,2) ===> 6
Ievadiet matricas elememtu a(2,0) ===> 6
Ievadiet matricas elememtu a(2,1) ===> 6
Ievadiet matricas elememtu a(2,2) ===> 6

Pirmā ievadīta matrica:
    1    2    3
    4    5    6
    7    0    0

Otrā ievadīta matrica:
    6    6    6
    6    6    6
    6    6    6

Abu matricu atbilstošās elementu reizinājums:
    6   12   18
   24   30   36
   42    0    0

Divu matricu reizinājums:
   36   36   36
   90   90   90
   42   42   42
```

6)

```
Ievadiet 1.matricas rindu skaitu ===> 12.2
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas rindu skaitu ===> -12.2
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas rindu skaitu ===> abc
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas rindu skaitu ===> 3
Ievadiet 1.matricas kolonnu skaitu ===> -1
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> -12.2
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> 12.5
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> pieci
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> 5
Ievadiet matricas elememtu a(0,0) ===> -1.2
Kļūda! Ievadītais elements nav vesels skaitlis!
Ievadiet matricas elememtu a(0,0) ===> pieci
Kļūda! Ievadītais elements nav vesels skaitlis!
Ievadiet matricas elememtu a(0,0) ===> 5
Ievadiet matricas elememtu a(0,1) ===> 5
Ievadiet matricas elememtu a(0,2) ===> 5
Ievadiet matricas elememtu a(0,3) ===> 5
Ievadiet matricas elememtu a(0,4) ===> 5
Ievadiet matricas elememtu a(1,0) ===> 5
Ievadiet matricas elememtu a(1,1) ===> 5
Ievadiet matricas elememtu a(1,2) ===> 5
Ievadiet matricas elememtu a(1,3) ===> 5
Ievadiet matricas elememtu a(1,4) ===> 5
Ievadiet matricas elememtu a(2,0) ===> 5
Ievadiet matricas elememtu a(2,1) ===> 5
Ievadiet matricas elememtu a(2,2) ===> 5
Ievadiet matricas elememtu a(2,3) ===> -1222.2
Kļūda! Ievadītais elements nav vesels skaitlis!
Ievadiet matricas elememtu a(2,3) ===> 0
Ievadiet matricas elememtu a(2,4) ===> 0
Ievadiet 2.matricas rindu skaitu ===> 0
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 2.matricas rindu skaitu ===> -1
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 2.matricas rindu skaitu ===> 1
Ievadiet 2.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 10

Pirmā ievadīta matrica:
    5    5    5    5    5
    5    5    5    5    5
    5    5    5    0    0

Otrā ievadīta matrica:
   10

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Kļūda! Šādas matricas nevar sareizināt, jo 1.matricas kolonnu skaits nav vienāds ar 2.matricas rindas skaitu.
```

7)

```
Ievadiet 1.matricas rindu skaitu ===> 12.2
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas rindu skaitu ===> -12.2
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas rindu skaitu ===> abc
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas rindu skaitu ===> 3
Ievadiet 1.matricas kolonnu skaitu ===> -1
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> -12.2
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> 12.5
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> pieci
Kļūda! Matricas kolonnu skaitam jābut naturālam skaitlim!
Ievadiet 1.matricas kolonnu skaitu ===> 5
Ievadiet matricas elememtu a(0,0) ===> -1.2
Kļūda! Ievadītais elements nav vesels skaitlis!
Ievadiet matricas elememtu a(0,0) ===> pieci
Kļūda! Ievadītais elements nav vesels skaitlis!
Ievadiet matricas elememtu a(0,0) ===> 5
Ievadiet matricas elememtu a(0,1) ===> 5
Ievadiet matricas elememtu a(0,2) ===> 5
Ievadiet matricas elememtu a(0,3) ===> 5
Ievadiet matricas elememtu a(0,4) ===> 5
Ievadiet matricas elememtu a(1,0) ===> 5
Ievadiet matricas elememtu a(1,1) ===> 5
Ievadiet matricas elememtu a(1,2) ===> 5
Ievadiet matricas elememtu a(1,3) ===> 5
Ievadiet matricas elememtu a(1,4) ===> 5
Ievadiet matricas elememtu a(2,0) ===> 5
Ievadiet matricas elememtu a(2,1) ===> 5
Ievadiet matricas elememtu a(2,2) ===> 5
Ievadiet matricas elememtu a(2,3) ===> -1222.42
Kļūda! Ievadītais elements nav vesels skaitlis!
Ievadiet matricas elememtu a(2,3) ===> 0
Ievadiet matricas elememtu a(2,4) ===> 0

Ievadiet 2.matricas rindu skaitu ===> 0
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 2.matricas rindu skaitu ===> 0
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 2.matricas rindu skaitu ===> -1
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 2.matricas rindu skaitu ===> 1
Ievadiet 2.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 10

Pirmā ievadīta matrica:
    5    5    5    5    5
    5    5    5    5    5
    5    5    5    0    0

Otrā ievadīta matrica:
   10

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Kļūda! Šādas matricas nevar sareizināt, jo 1.matricas kolonnu skaits nav vienāds ar 2.matricas rindas skaitu.
```

8)

```
Ievadiet 1.matricas rindu skaitu ===> 2
Ievadiet 1.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 3
Ievadiet matricas elememtu a(1,0) ===> 4
Ievadiet matricas elememtu a(1,1) ===> 3
Ievadiet matricas elememtu a(1,2) ===> 2

Ievadiet 2.matricas rindu skaitu ===> 3
Ievadiet 2.matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(1,0) ===> 3
Ievadiet matricas elememtu a(1,1) ===> 5
Ievadiet matricas elememtu a(2,0) ===> 6
Ievadiet matricas elememtu a(2,1) ===> 4

Pirmā ievadīta matrica:
    1    2    3
    4    3    2

Otrā ievadīta matrica:
    1    2
    3    5
    6    4

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Divu matricu reizinājums:
   25   24
   25   31
```

9)

```
Ievadiet 1.matricas rindu skaitu ===> 1
Ievadiet 1.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 5

Ievadiet 2.matricas rindu skaitu ===> 1
Ievadiet 2.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 0

Pirmā ievadīta matrica:
     5

Otrā ievadīta matrica:
     0

Abu matricu atbilstošās elementu reizinājums:
     0

Divu matricu reizinājums:
     0
```

10)

```
Ievadiet 1.matricas rindu skaitu ===> 1
Ievadiet 1.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 5

Ievadiet 2.matricas rindu skaitu ===> 1
Ievadiet 2.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 6

Pirmā ievadīta matrica:
    5

Otrā ievadīta matrica:
    6

Abu matricu atbilstošās elementu reizinājums:
   30

Divu matricu reizinājums:
   30
```

11)

```
Ievadiet 1.matricas rindu skaitu ===> 2
Ievadiet 1.matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 0
Ievadiet matricas elememtu a(0,1) ===> 0
Ievadiet matricas elememtu a(1,0) ===> 0
Ievadiet matricas elememtu a(1,1) ===> 0

Ievadiet 2.matricas rindu skaitu ===> 2
Ievadiet 2.matricas kolonnu skaitu ===> 2
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 5
Ievadiet matricas elememtu a(1,0) ===> 3
Ievadiet matricas elememtu a(1,1) ===> 6

Pirmā ievadīta matrica:
    0    0
    0    0

Otrā ievadīta matrica:
    1    5
    3    6

Abu matricu atbilstošās elementu reizinājums:
    0    0
    0    0

Divu matricu reizinājums:
    0    0
    0    0
```

12)

```
Ievadiet 1.matricas rindu skaitu ===> 2
Ievadiet 1.matricas kolonnu skaitu ===> 4
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 5
Ievadiet matricas elememtu a(0,2) ===> 6
Ievadiet matricas elememtu a(0,3) ===> 8
Ievadiet matricas elememtu a(1,0) ===> 0
Ievadiet matricas elememtu a(1,1) ===> 0
Ievadiet matricas elememtu a(1,2) ===> 0
Ievadiet matricas elememtu a(1,3) ===> 0

Ievadiet 2.matricas rindu skaitu ===> 0
Kļūda! Matricas rindu skaitam jābut naturālam skaitlim!
Ievadiet 2.matricas rindu skaitu ===> 1
Ievadiet 2.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 2

Pirmā ievadīta matrica:
    1    5    6    8
    0    0    0    0

Otrā ievadīta matrica:
    2

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Kļūda! Šādas matricas nevar sareizināt, jo 1.matricas kolonnu skaits nav vienāds ar 2.matricas rindas skaitu.
```

13)

```
Ievadiet 1.matricas rindu skaitu ===> 2
Ievadiet 1.matricas kolonnu skaitu ===> 1
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(1,0) ===> 2

Ievadiet 2.matricas rindu skaitu ===> 3
Ievadiet 2.matricas kolonnu skaitu ===> 6
Ievadiet matricas elememtu a(0,0) ===> 5
Ievadiet matricas elememtu a(0,1) ===> 1
Ievadiet matricas elememtu a(0,2) ===> 4
Ievadiet matricas elememtu a(0,3) ===> 2
Ievadiet matricas elememtu a(0,4) ===> 6
Ievadiet matricas elememtu a(0,5) ===> 8
Ievadiet matricas elememtu a(1,0) ===> 2
Ievadiet matricas elememtu a(1,1) ===> 6
Ievadiet matricas elememtu a(1,2) ===> 9
Ievadiet matricas elememtu a(1,3) ===> 1
Ievadiet matricas elememtu a(1,4) ===> 2
Ievadiet matricas elememtu a(1,5) ===> 5
Ievadiet matricas elememtu a(2,0) ===> 8
Ievadiet matricas elememtu a(2,1) ===> 7
Ievadiet matricas elememtu a(2,2) ===> 2
Ievadiet matricas elememtu a(2,3) ===> 3
Ievadiet matricas elememtu a(2,4) ===> 6
Ievadiet matricas elememtu a(2,5) ===> 98

Pirmā ievadīta matrica:
    1
    2

Otrā ievadīta matrica:
    5    1    4    2    6    8
    2    6    9    1    2    5
    8    7    2    3    6   98

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Kļūda! Šādas matricas nevar sareizināt, jo 1.matricas kolonnu skaits nav vienāds ar 2.matricas rindas skaitu.
```

14)

```
Ievadiet 1.matricas rindu skaitu ===> 2
Ievadiet 1.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 3
Ievadiet matricas elememtu a(1,0) ===> 4
Ievadiet matricas elememtu a(1,1) ===> 5
Ievadiet matricas elememtu a(1,2) ===> 6

Ievadiet 2.matricas rindu skaitu ===> 3
Ievadiet 2.matricas kolonnu skaitu ===> 8
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 3
Ievadiet matricas elememtu a(0,3) ===> 4
Ievadiet matricas elememtu a(0,4) ===> 5
Ievadiet matricas elememtu a(0,5) ===> 6
Ievadiet matricas elememtu a(0,6) ===> 7
Ievadiet matricas elememtu a(0,7) ===> 8
Ievadiet matricas elememtu a(1,0) ===> 9
Ievadiet matricas elememtu a(1,1) ===> 10
Ievadiet matricas elememtu a(1,2) ===> 11
Ievadiet matricas elememtu a(1,3) ===> 12
Ievadiet matricas elememtu a(1,4) ===> 13
Ievadiet matricas elememtu a(1,5) ===> 14
Ievadiet matricas elememtu a(1,6) ===> 15
Ievadiet matricas elememtu a(1,7) ===> 16
Ievadiet matricas elememtu a(2,0) ===> 17
Ievadiet matricas elememtu a(2,1) ===> 18
Ievadiet matricas elememtu a(2,2) ===> 19
Ievadiet matricas elememtu a(2,3) ===> 20
Ievadiet matricas elememtu a(2,4) ===> 21
Ievadiet matricas elememtu a(2,5) ===> 22
Ievadiet matricas elememtu a(2,6) ===> 23
Ievadiet matricas elememtu a(2,7) ===> 24

Pirmā ievadīta matrica:
    1    2    3
    4    5    6

Otrā ievadīta matrica:
    1    2    3    4    5    6    7    8
    9   10   11   12   13   14   15   16
   17   18   19   20   21   22   23   24

Kļūda! Šādas matricas nevar sareizināt atbilstošus elementus.

Divu matricu reizinājums:
   70   76   82   88   94  100  106  112
  151  166  181  196  211  226  241  256
```

15)

```
Ievadiet 1.matricas rindu skaitu ===> 3
Ievadiet 1.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 2
Ievadiet matricas elememtu a(0,1) ===> 2
Ievadiet matricas elememtu a(0,2) ===> 2
Ievadiet matricas elememtu a(1,0) ===> 2
Ievadiet matricas elememtu a(1,1) ===> 2
Ievadiet matricas elememtu a(1,2) ===> 2
Ievadiet matricas elememtu a(2,0) ===> 2
Ievadiet matricas elememtu a(2,1) ===> 2
Ievadiet matricas elememtu a(2,2) ===> 2

Ievadiet 2.matricas rindu skaitu ===> 3
Ievadiet 2.matricas kolonnu skaitu ===> 3
Ievadiet matricas elememtu a(0,0) ===> 1
Ievadiet matricas elememtu a(0,1) ===> 1
Ievadiet matricas elememtu a(0,2) ===> 1
Ievadiet matricas elememtu a(1,0) ===> 1
Ievadiet matricas elememtu a(1,1) ===> 1
Ievadiet matricas elememtu a(1,2) ===> 1
Ievadiet matricas elememtu a(2,0) ===> 1
Ievadiet matricas elememtu a(2,1) ===> 1
Ievadiet matricas elememtu a(2,2) ===> 1

Pirmā ievadīta matrica:
    2    2    2
    2    2    2
    2    2    2

Otrā ievadīta matrica:
    1    1    1
    1    1    1
    1    1    1

Abu matricu atbilstošās elementu reizinājums:
    2    2    2
    2    2    2
    2    2    2

Divu matricu reizinājums:
    6    6    6
    6    6    6
    6    6    6
```

PU1. uzdevums

Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) dalīšanas ar atlikumu algoritmu. Jāveic ievaddatu korektuma pārbaude!

Kods:

```
# Programmas nosaukums: Lielo skaitļu dalīšana ar atlikumu

# 1.Papilduzdevums (1MPR08_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē lielo naturālo skaitļu (vismaz ar 50 cipariem) dalīšanas ar atlikumu algoritmu.

# Jāveic ievaddatu korektuma pārbaude!

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import math



def dalit_lielus_skaitlus(s, n):

    # Funkcija atgriež tuple (dalījums, atlikums) no s / n. Dalījums ir int un atlikums ir int.

    # Jāizmanto lielo skaitļu dalīšanai.

    # Dalā pēc "cipariem".

    # s - str, dalāmais

    # n - str, dalītājs

    n = int(n)

    galva = 0

    atlikums = 0

    dalijums_sv = ""


    for i in range(len(s)):


        cipars = int(s[i])

        tmp = cipars + atlikums * galva
```

```
            dalijums_sv = dalijums_sv + str(tmp // n)

            if tmp % n == 0:
                galva = 0
                atlikums = 0
            else:
                atlikums = tmp % n
                galva = 10

        dalijums = nodzest_liekas_nulles(dalijums_sv)
        dalijums = int(dalijums)

        return (dalijums, atlikums)



def nodzest_liekas_nulles(s):
    # Nodzes liekas nulles sākumā
    # s - simbolu virkne
    for i in range(len(s)):
        if s[i] != '0':
            return s[i:]
    return '0'



def is_natural_long(s):
    # Pārbauda vai simbolu virkne reprezentē naturālo skaitli, vai nē.
    # Atgriež True, ja virkne reprezentē naturālo skaitli.
    # Atgriež False, ja nereprezentē naturālo skaitli.
    # s - pārbaudama simbolu virkne

    # Noņema no virknes visas sākuma vai beigu atstarpes
```

```
    s = s.strip()

    # Pārbauda, vai virkne ir tukša
    if len(s) == 0 or (len(s) == 1 and s == "0"):
        return False

    # Iet cikliski cauri katrām simbolam simbolu virknē (string'ā)
    for c in s:
        # Ja kāda rakstzīme nav cipars, virkne neatspoguļo naturālu skaitli. return False
        if not c.isdigit():
            return False

    # Virkne atspoguļo naturālu skaitli, ja ietu cauri ciklas netika pamanīts not .isdigit()
    return True



def is_natural_or_zero_long(s):
    # Pārbauda vai simbolu virkne reprezentē naturālo skaitli vai 0, vai nē.
    # Atgriež True, ja virkne reprezentē naturālo skaitli.
    # Atgriež False, ja nereprezentē naturālo skaitli.
    # s - pārbaudama simbolu virkne

    # Noņema no virknes visas sākuma vai beigu atstarpes
    s = s.strip()

    if not s.isdigit():
        return False

    if len(s) == 1 and s == "0":
        return True
```

```
    # Iet cikliski cauri katrām simbolam simbolu virknē (string'ā)
    for c in s:
        # Ja kāda rakstzīme nav cipars, virkne neatspoguļo naturālu skaitli. return False
        if not c.isdigit():
            return False

    # Virkne atspoguļo naturālu skaitli, ja ietu cauri ciklas netika pamanīts not .isdigit()
    return True



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



dalamais = input("Ievadiet dalāmo ==> ")
while is_natural_or_zero_long(dalamais) == False:
    dalamais = input("Kļūda! Ievadiet naturālo dalāmo vai nulle ==> ")

dalitajs = input("Ievadiet dalītāju ==> ")
while is_natural_long(dalitajs) == False:
    dalitajs = input("Kļūda! Ievadiet naturālo dalītāju ==> ")

rezultats = dalit_lielus_skaitlus(dalamais, dalitajs)

print("\nNepilnais dalījums:")
print(rezultats[0])
print("\nAtlikums:")
print(rezultats[1])
```

Testa piemēri:

1)

```
Ievadiet dalāmo ==> 5
Ievadiet dalītāju ==> 2

Nepilnais dalījums:
2

Atlikums:
1
```

2)

```
Ievadiet dalāmo ==> 6
Ievadiet dalītāju ==> 2

Nepilnais dalījums:
3

Atlikums:
0
```

3)

```
Ievadiet dalāmo ==> 7
Ievadiet dalītāju ==> 2

Nepilnais dalījums:
3

Atlikums:
1
```

4)

```
Ievadiet dalāmo ==> 8
Ievadiet dalītāju ==> 2

Nepilnais dalījums:
4

Atlikums:
0
```

5)

```
Ievadiet dalāmo ==> 4641644641644641615166518964189489496 1
Ievadiet dalītāju ==> 161856418654156485964865 4

Nepilnais dalījums:
28677544457242

Atlikums:
95627759889046064904269 3
```

6)

```
Ievadiet dalāmo ==> 1000
Ievadiet dalītāju ==> 10

Nepilnais dalījums:
100

Atlikums:
0
```

7)

```
Ievadiet dalāmo ==> 0
Ievadiet dalītāju ==> 0
Kļūda! Ievadiet naturālo dalītāju ==> 1

Nepilnais dalījums:
0

Atlikums:
0
```

8)

```
Ievadiet dalāmo ==> 0
Ievadiet dalītāju ==> 165165161

Nepilnais dalījums:
0

Atlikums:
0
```

9)

```
Ievadiet dalāmo ==> 55555
Ievadiet dalītāju ==> 55555

Nepilnais dalījums:
1

Atlikums:
0
```

10)

```
Ievadiet dalāmo ==> 555555
Ievadiet dalītāju ==> 66666666666666666

Nepilnais dalījums:
0

Atlikums:
555555
```

11)

```
Ievadiet dalāmo ==> labi
Kļūda! Ievadiet naturālo dalāmo vai nulle ==> -1
Kļūda! Ievadiet naturālo dalāmo vai nulle ==> pieci
Kļūda! Ievadiet naturālo dalāmo vai nulle ==> 5
Ievadiet dalītāju ==> 5

Nepilnais dalījums:
1

Atlikums:
0
```

12)

```
Ievadiet dalāmo ==> 4869999999999999999999999999999999999564416415641648654646416164568346341434856464641564111111111111111111111111111111111111111111111111111111111111111111111111111111111
Ievadiet dalītāju ==> 614555555555555555555555555555555555555555555555555555555555555555555

Nepilnais dalījums:
7924425962755378774181883926957150604968314543622281303890389709115016578384292525523042471419207157

Atlikums:
42891649063635345711835494660949508342543546829180846134706344003976
```

13)

```
Ievadiet dalāmo ==> 46666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
Ievadiet dalītāju ==> 6455555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555554444444444444

Nepilnais dalījums:
7228915

Atlikums:
4277777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777785809905555558768406
```

14)

```
Ievadiet dalāmo ==> 2132333235641651651564856456456415645645645645645656456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456
Ievadiet dalītāju ==> 456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456456565645645645651

Nepilnais dalījums:
467

Atlikums:
6815839899999999132048048047639939939939939940048048048048048048048048048048048048048048048048048048048048048048048048048048048048048048048048048048042999399399399937439
```

PU2. uzdevums

Sastādīt programmu, kas realizē kvadrātsaknes izvilkšanas algoritmu lielajiem naturālajiem skaitļiem (vismaz ar 50 cipariem) ar precizitāti līdz veselam skaitlim. Jāveic ievaddatu korektuma pārbaude!

Kods:

```
# Programmas nosaukums: Lielo skaitļu kvadrātsaknes izvilkšanas algoritms

# 2.Papilduzdevums (1MPR08_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē kvadrātsaknes izvilkšanas algoritmu lielajiem naturālajiem skaitļiem (vismaz ar 50 cipariem) ar precizitāti līdz veselam skaitlim.

# Jāveic ievaddatu korektuma pārbaude!

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



# import math  # Testēšanai



def long_sqrt(skaitlis):

    # Funkciju, kas kā ievadi izmanto nenegatīvu veselu skaitli un atgriež tā veselā skaitļa kvadrātsakni

    # skaitlis - int, skaitlis, no kura vajag atrast kvadrātsakni

    # Atgriež sqrt(skaitlis) kā int


    # Konvertējat ievadīto veselo nenegatīvo skaitli par simbolu virkni un saglabājam to mainīgajā

    sk_sv = str(skaitlis)  # sk_sv - skaitlis_simbolu virkne

    # Saglabājam ievadītā veselā skaitļa simbolu virknes garumu

    len_sk = len(sk_sv)  # slen_skaitlis


    # Ja ievadītā veselā skaitļa simbolu virknes garums ir nepāra skaitlis, tad pievienojam sākuma nulli, lai padarītu to par pāru simbolu virkni (garums ir pāra skaitlis)

    if len_sk % 2 == 1:
```

```
    sk_sv = '0' + sk_sv  # Ievadītā veselā skaitļa simbolu virknei sākumam pievienojam '0'
    len_sk = len_sk + 1  # Palielinām ievadītā veselā skaitļa simbolu virknes garumu par 1

result = ''  # Tukša simbolu virkne, lai saglabātu kvadrātsaknes aprēķina rezultātu
atlikums = 0

# Sadalām ievadītā veselā skaitļa virknes pirmos divus ciparus [:2]
a = int(sk_sv[:2])

j = 1            # Atrodam pirmo "tuvinājumu"
while j * j <= a:  # Tas ir lai atrastu pirmo ciparu kvadrātsaknei
    j = j + 1      # Var arī atrast tā:
sakne = j - 1      # sakne = int(a**0.5) (lai nebūtu cikls)

result = result + str(sakne)
atlikums = a - sakne * sakne  # Atlikums
# Cikls pār katru otro "ciparu pāri"
for i in range(2, len_sk, 2):  # Stabiņveidā atrodam nākamo un nākamo un ... un pēdējo ciparu
    # Reizinām atlikumu ar 100 un pievienojiet nākamos divus veselā skaitļa ciparus, lai iegūtu dividendi
    a = atlikums * 100 + int(sk_sv[i:i + 2])
    cipars = 0
    temp = sakne * 20
    while (temp + cipars) * cipars <= a:
        cipars += 1
    cipars -= 1
    result += str(cipars)  # Rezultātam pievienojam iegūto ciparu
    atlikums = a - (temp + cipars) * cipars
    sakne = sakne * 10 + cipars

return int(result)
```

```
def is_natural_or_zero_long(s):
    # Pārbauda vai simbolu virkne reprezentē naturālo skaitli vai 0, vai nē.
    # Atgriež True, ja virkne reprezentē naturālo skaitli.
    # Atgriež False, ja nereprezentē naturālo skaitli.
    # s - pārbaudāma simbolu virkne

    # Noņema no virknes visas sākuma vai beigu atstarpes
    s = s.strip()

    if not s.isdigit():
        return False

    if len(s) == 1 and s == "0":
        return True

    # Iet cikliski cauri katrām simbolam simbolu virknē (string'ā)
    for c in s:
        # Ja kāda rakstzīme nav cipars, virkne neatspoguļo naturālu skaitli. return False
        if not c.isdigit():
            return False

    # Virkne atspoguļo naturālu skaitli, ja ietu cauri ciklas netika pamanīts not .isdigit()
    return True


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

```
n = input("Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> ")
while is_natural_or_zero_long(n) == False:
    n = input("Kļūda! Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> ")
n = int(n)


print("\nsqrt(" + str(n) + ")")
print("≈")
print(long_sqrt(n))
# math.sqrt(skaitlis) # Testēšanai


# Testēšanai
'''
for i in range(1, 10000000, 1):  # for i in range(1, 1000000, 1):
    print("sqrt(" + str(i) + ") =", math.floor(math.sqrt(i)))
    print(str(long_sqrt(i)) + " == " + str(math.floor(math.sqrt(i))))
    if long_sqrt(i) == math.floor(math.sqrt(i)):
        print("TRUE\n")
    else:
        print("FALSE\n")
'''
```

Testa piemēri:

1)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 11111111111111111111111111111111111111111111

sqrt(11111111111111111111111111111111111111111111)
≈
333333333333333333333
```

2)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 25

sqrt(25)
≈
5
```

3)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 12345

sqrt(12345)
≈
111
```

4)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 9999

sqrt(9999)
≈
99
```

5)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 666

sqrt(666)
≈
25
```

6)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 0

sqrt(0)
≈
0
```

7)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 1

sqrt(1)
≈
1
```

8)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 25

sqrt(25)
≈
5
```

9)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

sqrt(1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111)
≈
33333333333333333333333333333333333333333333333333333333333333333333
```

10)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666

sqrt(6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666)
≈
2581988897471611256786176933188266407221947803527727217725049177408988872795
```

11)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 987654321

sqrt(987654321)
≈
31426
```

12)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 225

sqrt(225)
≈
15
```

13)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 1125

sqrt(1125)
≈
33
```

14)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 625

sqrt(625)
≈
25
```

15)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 390625

sqrt(390625)
≈
625
```

16)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 11111

sqrt(11111)
≈
105
```

17)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 567890

sqrt(567890)
≈
753
```

18)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 55555

sqrt(55555)
≈
235
```

19)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 555555

sqrt(555555)
≈
745
```

20)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> pieci
Kļūda! Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> -5
Kļūda! Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> -2
Kļūda! Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 12.5
Kļūda! Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 12

sqrt(12)
≈
3
```

21)

```
Ievadiet naturālo skaitli vai 0 kvadrātsaknes vilkšanai ==> 33333333

sqrt(33333333)
≈
5773
```