1. uzdevums

Sastādīt programmu, kas atrod 5 naturālo skaitļu lielāko kopīgo dalītāju un mazāko kopīgo dalāmo. Skaitļus ievada lietotājs, bet aprēķini tiek veikti tikai tad, ja lietotājs ir ievadījis naturālus skaitļus, pretējā gadījumā tiek izvadīts atbilstošs paziņojums.

Kods:

```
# Programmas nosaukums: 1. uzd MPR14

# 1. uzdevums MPR14

# Uzdevuma formulējums: Sastādīt programmu, kas atrod 5 naturālo skaitļu lielāko kopīgo dalītāju un mazāko kopīgo dalāmo. Skaitļus ievada lietotājs, bet aprēķini tiek veikti tikai tad, ja lietotājs ir ievadījis naturālus skaitļus, pretējā gadījumā tiek izvadīts atbilstošs paziņojums.

# Versija 1.0


import math


# Divu skaitļu gcd atrāšana

def my_gcd(a,b):


   while b != 0 :

      c = a % b

      a = b

      b = c


   return a


# Divu skaitļu lcm atrāšana

# lcm (a, b) = a*b/(gcd (a,b))

def my_lcm(a,b):


   return a*b/my_gcd(a,b)


print("Programma atrod 5 naturālo skaitļu lielāko kopīgo dalītāju un mazāko kopīgo dalāmo.\nIevadiet naturālus skaitļus!\n")
```

```
x = input("Ievadi 1.skaitli ===> ") # skaitlis a
y = input("Ievadi 2.skaitli ===> ") # skaitlis b
z = input("Ievadi 3.skaitli ===> ")
w = input("Ievadi 4.skaitli ===> ")
q = input("Ievadi 5.skaitli ===> ")

#skaitļu parbaude vai viņi ir veseli pozitivie skaitļi (dalam ar 0 un sqrt()) tests.
M=0
while M<=3:

    if M==1 or M==2:
        #print("Viens no ievadītajiem skaitliem nav naturāls skaitlis!")
        x = input("Ievadi 1.skaitli ===> ") # skaitlis a
        y = input("Ievadi 2.skaitli ===> ") # skaitlis b
        z = input("Ievadi 3.skaitli ===> ")
        w = input("Ievadi 4.skaitli ===> ")
        q = input("Ievadi 5.skaitli ===> ")
    elif M==3:
        print("Jūs 3 reizes kļūdījāties. Beidzam sadarbību!")
        quit()

    try:
        x = int(x)
        a = 1/x
        b=math.sqrt(x)

        y = int(y)
        c = 1/y
        d = math.sqrt(y)

        z = int(z)
```

```
        e = 1/z
        f = math.sqrt(z)

        w = int(w)
        g = 1/w
        h = math.sqrt(w)

        q = int(q)
        k = 1/q
        v = math.sqrt(q)

    except:
        M=M+1
        print("\nViens no ievadītajiem skaitļiem nav naturāls skaitlis!\n")



    else:
        #noteiksim gcd no 5 skaitliem
        t=my_gcd(x,y)
        u=my_gcd(t,z)
        l=my_gcd(u,w)
        v=my_gcd(l,q)
        #print(v)
        #print("gcd(" + str(x) + ", " + str(y) + ", " + str(z) + ", " + str(w) + ", " + str(q) + ") = " +
str(v))
        print("LKD(" + str(x) + ", " + str(y) + ", " + str(z) + ", " + str(w) + ", " + str(q) + ") = " + str(v))

        #noteiksim lcm no 5 skaitliem
        a=my_lcm(x,y)
        s=my_lcm(a,z)
        d=my_lcm(s,w)
        g=my_lcm(d,q)
```

```
    #print(g)

    # print("lcm(" + str(x) + ", " + str(y) + ", " + str(z) + ", " + str(w) + ", " + str(q) + ") = " +
str(g))

    print("MKD(" + str(x) + ", " + str(y) + ", " + str(z) + ", " + str(w) + ", " + str(q) + ") = " +
str(g))

    break
```

Testa piemēri:

1)

```
Programma atrod 5 naturālo skaitļu lielāko kopīgo dalītāju un mazāko kopīgo dalāmo.
Ievadiet naturālus skaitļus!

Ievadi 1.skaitli ===> s
Ievadi 2.skaitli ===> s
Ievadi 3.skaitli ===> 1
Ievadi 4.skaitli ===> 2
Ievadi 5.skaitli ===> 3

Viens no ievadītajiem skaitļiem nav naturāls skaitlis!

Ievadi 1.skaitli ===> 4
Ievadi 2.skaitli ===> 0.24
Ievadi 3.skaitli ===> 1
Ievadi 4.skaitli ===> 2
Ievadi 5.skaitli ===> 3

Viens no ievaditajiem skaitļiem nav naturāls skaitlis!

Ievadi 1.skaitli ===> 1
Ievadi 2.skaitli ===> 2
Ievadi 3.skaitli ===> 3
Ievadi 4.skaitli ===> 4
Ievadi 5.skaitli ===> 5
LKD(1, 2, 3, 4, 5) = 1
MKD(1, 2, 3, 4, 5) = 60.0
```

2)

```
Programma atrod 5 naturālo skaitļu lielāko kopīgo dalītāju un mazāko kopīgo dalāmo.
Ievadiet naturālus skaitļus!

Ievadi 1.skaitli ===> 0.24
Ievadi 2.skaitli ===> 1
Ievadi 3.skaitli ===> 2
Ievadi 4.skaitli ===> 3
Ievadi 5.skaitli ===> 4

Viens no ievadītajiem skaitļiem nav naturāls skaitlis!

Ievadi 1.skaitli ===> ads
Ievadi 2.skaitli ===> 1
Ievadi 3.skaitli ===> 2
Ievadi 4.skaitli ===> 3
Ievadi 5.skaitli ===> 4

Viens no ievadītajiem skaitļiem nav naturāls skaitlis!

Ievadi 1.skaitli ===> 0
Ievadi 2.skaitli ===> 1
Ievadi 3.skaitli ===> 2
Ievadi 4.skaitli ===> 3
Ievadi 5.skaitli ===> 4

Viens no ievadītajiem skaitļiem nav naturāls skaitlis!

Jūs 3 reizes kļūdījāties. Beidzam sadarbību!
```

3)

```
Programma atrod 5 naturālo skaitļu lielāko kopīgo dalītāju un mazāko kopīgo dalāmo.
Ievadiet naturālus skaitļus!

Ievadi 1.skaitli ===> 10
Ievadi 2.skaitli ===> 20
Ievadi 3.skaitli ===> 30
Ievadi 4.skaitli ===> 40
Ievadi 5.skaitli ===> 50
LKD(10, 20, 30, 40, 50) = 10
MKD(10, 20, 30, 40, 50) = 600.0
```

2. uzdevums

Sastādīt programmu, kas nodrukā visus “draudzīgos” pirmskaitļu pārus no 1 līdz N. Par draudzīgu pirmskaitļu pāri tādus divus pirmskaitļus, kuru starpība ir 2. Naturālo skaitli N ievada lietotājs un ievades procesā drīkst kļūdīties ne vairāk kā 3 reizes.

Kods:

```
# Programmas nosaukums: 2. uzd MPR14

# 2. uzdevums MPR14

# Uzdevuma formulējums: Sastādīt programmu, kas nodrukā visus “draudzīgos” pirmskaitļu pārus no 1 līdz N. Par draudzīgu pirmskaitļu pāri tādus divus pirmskaitļus, kuru starpība ir 2. Naturālo skaitli N ievada lietotājs un ievades procesā drīkst kļūdīties ne vairāk kā 3 reizes.

# Versija 1.0
```

```
import math


def vaipirmskaitlis (n):
    M = round(math.sqrt(n))+1
    for i in range(2, M):
        if n % i == 0:
            return False
    return True


def draugi(x,y):
    if abs(x-y)==2:
        return True
    else:
        return False



print("Programma nodrukā visus “draudzīgos” pirmskaitļu pārus no 1 līdz N.\nPar draudzīgu pirmskaitļu pāri tādus divus pirmskaitļus, kuru starpība ir 2.\n")



M=0
while M<=3:


    if M==0:
        x=input("Ievadiet naturālo skaitli N => ")
    elif M==1 or M==2:
        x=input("Tas nav naturāls skaitlis. Ievadiet naturālo skaitli N => ")
    elif M==3:
        x=input("Jūs 3 reizes kļūdījāties. Beidzam sadarbību!")
        quit()
```

```
try:
    x = int(x)
    a = 1/x
    b=math.sqrt(x)

except:
    M=M+1

else:
    if x<5:
        print("Pāru nav")
        break
    else:

        sv=""
        b = 3
        for i in range (5, x+1, 2):
            if vaipirmskaitlis(i):
                a = b
                b = i
                if draugi (a,b):
                    sv = sv + (str(a) + " " + str(b)) + "\n"

        print(sv)
        break
```

Testa piemēri:

1)

```
Programma nodrukā visus “draudzīgos” pirmskaitļu pārus no 1 līdz N.
Par draudzīgu pirmskaitļu pāri tādus divus pirmskaitļus, kuru starpība ir 2.

Ievadiet naturālo skaitli N => 0
Tas nav naturāls skaitlis. Ievadiet naturālo skaitli N => -1
Tas nav naturāls skaitlis. Ievadiet naturālo skaitli N => asf
Jūs 3 reizes kļūdījāties. Beidzam sadarbību!
```

2)

```
Programma nodrukā visus “draudzīgos” pirmskaitļu pārus no 1 līdz N.
Par draudzīgu pirmskaitļu pāri tādus divus pirmskaitļus, kuru starpība ir 2.

Ievadiet naturālo skaitli N => 50
3 5
5 7
11 13
17 19
29 31
41 43
```

3)

```
Programma nodrukā visus “draudzīgos” pirmskaitļu pārus no 1 līdz N.
Par draudzīgu pirmskaitļu pāri tādus divus pirmskaitļus, kuru starpība ir 2.

Ievadiet naturālo skaitli N => 0
Tas nav naturāls skaitlis. Ievadiet naturālo skaitli N => -100
Tas nav naturāls skaitlis. Ievadiet naturālo skaitli N => 132
3 5
5 7
11 13
17 19
29 31
41 43
59 61
71 73
101 103
107 109
```

3. uzdevums (un PU1 – ar datu korektuma pārbaudi)

Sastādīt programmu, kas nosaka, vai dotais piecstūris ir vai nav izliekts. Ja piecstūris ir izliekts, tā aprēķina tā laukumu. Piecstūra virsotņu koordinātas secīgi pulksteņa rādītajā virzienā ievada lietotājs.

Kods:

Programmas nosaukums: 3. uzd MPR14

3. uzdevums MPR14

```
# Uzdevuma formulējums: Sastādīt programmu, kas nosaka, vai dotais piecstūris ir vai nav izliekts. Ja piecstūris ir izliekts, tā aprēķina tā laukumu. Piecstūra virsotņu koordinātas secīgi pulksteņa rādītajā virzienā ievada lietotājs.

# Versija 1.0


import math


def line_and_check(x1,y1,x3,y3,x2,y2,x4,y4):

    a = y3-y1

    b = x1-x3

    c = x3*y1-x1*y3


    z1=a*x2+b*y2+c

    z3=a*x4+b*y4+c


    if (z1*z3>0) :

        return False

        #print("Punkti ir vienā pusē taisnei.")

    else :

        return True

        #print("Punkti nav vienā pusē taisnei.")


# x*(y3-y1)+y*(x1-x3)+x3*y1-x1*y3=0


print("Programma nosaka, vai dotais piecstūris ir vai nav izliekts.\nJa piecstūris ir izliekts, tā aprēķina tā laukumu.\nPiecstūra virsotņu koordinātas secīgi pulksteņa rādītajā virzienā ievada lietotājs.\n")

x1 = input("Ievadi x1 ===> ")

y1 = input("Ievadi y1 ===> ")


x2 = input("Ievadi x2 ===> ")

y2 = input("Ievadi y2 ===> ")


x3 = input("Ievadi x3 ===> ")

y3 = input("Ievadi y3 ===> ")
```

```
x4 = input("Ievadi x4 ===> ")
y4 = input("Ievadi y4 ===> ")

x5 = input("Ievadi x5 ===> ")
y5 = input("Ievadi y5 ===> ")

M=1
while M<3:
    try:
        x1 = float(x1)
        y1 = float(y1)

        x2 = float(x2)
        y2 = float(y2)

        x3 = float(x3)
        y3 = float(y3)

        x4 = float(x4)
        y4 = float(y4)

        x5 = float(x5)
        y5 = float(y5)

    except:
        M=M+1
        print("\nKaut viena koordināta tika nekorekti ievadīta. Ievadiet realus skaitļus!\n")
        x1 = input("Ievadi x1 ===> ")
        y1 = input("Ievadi y1 ===> ")

        x2 = input("Ievadi x2 ===> ")
        y2 = input("Ievadi y2 ===> ")
```

```
        x3 = input("Ievadi x3 ===> ")
        y3 = input("Ievadi y3 ===> ")

        x4 = input("Ievadi x4 ===> ")
        y4 = input("Ievadi y4 ===> ")

        x5 = input("Ievadi x5 ===> ")
        y5 = input("Ievadi y5 ===> ")

    else:

        if ((line_and_check(x1,y1,x3,y3,x2,y2,x4,y4)) and line_and_check(x1,y1,x3,y3,x2,y2,x5,y5) and
            line_and_check(x1,y1,x4,y4,x2,y2,x5,y5) and line_and_check(x1,y1,x4,y4,x3,y3,x5,y5) and
            line_and_check(x2,y2,x4,y4,x3,y3,x1,y1) and line_and_check(x2,y2,x4,y4,x3,y3,x5,y5) and
            line_and_check(x2,y2,x5,y5,x1,y1,x3,y3) and line_and_check(x2,y2,x5,y5,x1,y1,x4,y4) and
            line_and_check(x3,y3,x5,y5,x4,y4,x2,y2) and line_and_check(x3,y3,x5,y5,x4,y4,x1,y1) and
            line_and_check(x3,y3,x1,y1,x2,y2,x5,y5) and line_and_check(x3,y3,x1,y1,x2,y2,x4,y4) and
            line_and_check(x4,y4,x2,y2,x1,y1,x3,y3) and line_and_check(x4,y4,x2,y2,x5,y5,x3,y3) and
            line_and_check(x5,y5,x2,y2,x1,y1,x3,y3) and line_and_check(x5,y5,x2,y2,x1,y1,x4,y4)):

            print("\nPiecstūris ir izliekts")
            S1 = 0.5*((x2-x1)*(y3-y1)-(x3-x1)*(y2-y1))
            S2 = 0.5*((x3-x1)*(y5-y1)-(x5-x1)*(y3-y1))
            S3 = 0.5*((x3-x5)*(y4-y5)-(x4-x5)*(y3-y5))
            S = S1 + S2 + S3
            print("S = " + str(abs(S)))
            quit()
        else:
            print("\nPiecstūris ir ieliekts. Programma nevar aprēķināt ieliekta piecstūra laukumu.")
            quit()

print("\nJūs 3 reizes kļūdījāties. Beidzam sadarbību!")
```

Testa piemēri:

1)

```
Programma nosaka, vai dotais piecstūris ir vai nav izliekts.
Ja piecstūris ir izliekts, tā aprēķina tā laukumu.
Piecstūra virsotņu koordinātas secīgi pulksteņa rādītajā virzienā ievada lietotājs.

Ievadi x1 ===> 0
Ievadi y1 ===> 0
Ievadi x2 ===> 0
Ievadi y2 ===> 3
Ievadi x3 ===> 4
Ievadi y3 ===> 3
Ievadi x4 ===> 2
Ievadi y4 ===> 1
Ievadi x5 ===> 7
Ievadi y5 ===> 0

Piecstūris ir ieliekts. Programma nevar aprēķināt ieliekta piecstūra laukumu.
```

2)

```
Programma nosaka, vai dotais piecstūris ir vai nav izliekts.
Ja piecstūris ir izliekts, tā aprēķina tā laukumu.
Piecstūra virsotņu koordinātas secīgi pulksteņa rādītajā virzienā ievada lietotājs.

Ievadi x1 ===> 0
Ievadi y1 ===> 0
Ievadi x2 ===> -1
Ievadi y2 ===> 2
Ievadi x3 ===> 1
Ievadi y3 ===> 5
Ievadi x4 ===> 6
Ievadi y4 ===> 4
Ievadi x5 ===> 4
Ievadi y5 ===> 0

Piecstūris ir izliekts
S = 24.5
```

3)

```
Programma nosaka, vai dotais piecstūris ir vai nav izliekts.
Ja piecstūris ir izliekts, tā aprēķina tā laukumu.
Piecstūra virsotņu koordinātas secīgi pulksteņa rādītajā virzienā ievada lietotājs.

Ievadi x1 ===> hboh
Ievadi y1 ===> 1
Ievadi x2 ===> 2
Ievadi y2 ===> 3
Ievadi x3 ===> 4
Ievadi y3 ===> 5
Ievadi x4 ===> 6m
Ievadi y4 ===> afa
Ievadi x5 ===> 3
Ievadi y5 ===> 5

Kaut viena koordināta tika nekorekti ievadīta. Ievadiet realus skaitļus!

Ievadi x1 ===> olhghad
Ievadi y1 ===> 2
Ievadi x2 ===> 1
Ievadi y2 ===> 1
Ievadi x3 ===> 3
Ievadi y3 ===> 4
Ievadi x4 ===> 5
Ievadi y4 ===> 6
Ievadi x5 ===> 7
Ievadi y5 ===> 1

Kaut viena koordināta tika nekorekti ievadīta. Ievadiet realus skaitļus!

Ievadi x1 ===> ahpdapd
Ievadi y1 ===> 12123
Ievadi x2 ===> 3
Ievadi y2 ===> 1
Ievadi x3 ===> 2
Ievadi y3 ===> 2
Ievadi x4 ===> 4
Ievadi y4 ===> 6
Ievadi x5 ===> ds
Ievadi y5 ===> 2

Jūs 3 reizes kļūdījāties. Beidzam sadarbību!
```

4. uzdevums

Sastādīt programmu, kas koordinātu plaknē uzzīmē funkcijas y = ax^4 + bx^3 + cx^2 + dx + e grafiku. Koeficientus ievada lietotājs. Lietotājam jāparedz atsevišķa procedūra koordinātu plaknes uzzīmēšanai un funkcijas grafika konstruēšanai, kā ar funkcija dotās funkcijas vērtības aprēķināšanai.

Kods:

```
# Programmas nosaukums: 4. uzd MPR14

# 4. uzdevums MPR14

# Uzdevuma formulējums: Sastādīt programmu, kas koordinātu plaknē uzzīmē funkcijas y = ax^4 + bx^3 + cx^2 + dx + e grafiku. Koeficientus ievada lietotājs. Lietotājam jāparedz atsevišķa procedūra koordinātu plaknes uzzīmēšanai un funkcijas grafika konstruēšanai, kā ar funkcija dotās funkcijas vērtības aprēķināšanai.

# Versija 1.0


import tkinter

from tkinter import ttk

from tkinter import *


def enable_button(): # Aktīve pogu "Parādīt funkciju"

    poga3['state'] = ACTIVE


def disable_button(): # Dizaktīve pogu "Parādīt funkciju"

    poga3['state'] = DISABLED


def paradit(): # funkcija koodinātu zīmēšanai


    enable_button()

    kanva.configure(bg='AliceBlue')


    kanva.create_line(150,350,850,350, arrow="last", fill="gray")

    kanva.create_text(847, 330, text="X", anchor = "nw")


    kanva.create_line(500, 0, 500, 700, arrow="first", fill="gray")

    kanva.create_text(507, 0, text= "Y", anchor="nw")


    # Y ass
```

```
    for i in range (175, 826, 25): #Y ASS
        kanva.create_line(i, 347, i, 353, fill="gray")

    for i in range(-13,0): # minusiem uz Y
        kanva.create_text(505, 341 - i*25, text = str(i), anchor = "nw", fill= "gray")

    for i in range(1,14): # minusiem uz X
        kanva.create_text(505, 341 - i*25, text = str(i), anchor = "nw", fill= "gray")

    # X ass

    for i in range (25, 676,25) : # X ASS
        kanva.create_line(497, i, 503, i, fill="gray")

    for i in range(-13,0): # minusiem uz X
        kanva.create_text(490 + i*25, 330, text = str(i), anchor = "nw", fill= "gray")

    for i in range(1, 14): # plusiem uz X
        kanva.create_text(496 + i*25, 330, text = str(i), anchor = "nw", fill= "gray")

def get_point(a,b,c,d,e,x):
    return a*x*x*x*x + b*x*x*x + c*x*x + d*x + e # return funkcijas vērību punktā x

def zimesana(x1,y1,x2,y2): # grafiku zimēšanai. Savieno divus punktus ar nogriežņi
    kanva.create_line(x1, y1, x2, y2)

def funkcijas_vertibas(a,b,c,d,e,x,y,x1,y1):

    for i in range (175, 825):
        x = (i - 500)/25
        y = get_point(a,b,c,d,e,x) # Funkcija
```

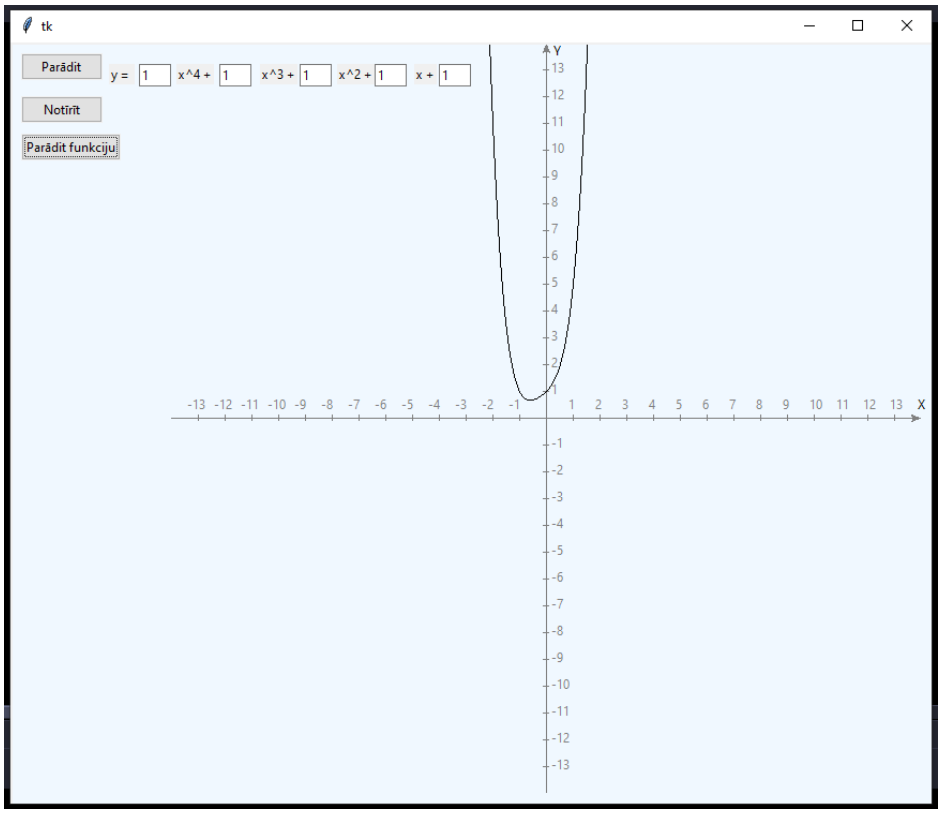
```
            x2 = i
            y2 = 350-y*25
            zimesana(x1,y1,x2,y2)
            x1=x2
            y1=y2

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

def funkcija_get(): # funkcija noteic pirmas funkcijas vērtības un paņēm no ievadisanas laukumiem ievaditus datus

    a = float(e1.get())
    b = float(e2.get())
    c = float(e3.get())
    d = float(e4.get())
    e = float(e5.get())

    x = (175 - 500)/25
    y = get_point(a,b,c,d,e,x) # Funkcija
    x1 = 175
    y1 = 350-y*25
    funkcijas_vertibas(a,b,c,d,e,x,y,x1,y1)

def notirit(): # funkcija satura dzēšanai

    kanva.configure(bg='white')
    kanva.delete("all")
    disable_button()

# Loga atribūti

logs = tkinter.Tk()
```

```
logs.geometry("860x710")

# Kanvas novietošana

kanva = tkinter.Canvas(logs, bg="white", height=710, width=860)
kanva.place(x=0, y=0)

# Pogas

poga1= ttk.Button(logs, text="Parādit", command=paradit)
poga1.place(x=10, y=10)

poga2 = ttk.Button(logs, text="Notīrīt", command=notirit)
poga2.place(x=10, y=50)

poga3= ttk.Button(logs, text="Parādit funkciju", state = 'disabled', command=funkcija_get)
poga3.place(x=10, y=85)

# Ievadisanas laukumi

e1 = ttk.Entry(logs)
e1.place(x=120, y=20, width=30)

e2 = ttk.Entry(logs)
e2.place(x=195, y=20, width=30)

e3 = ttk.Entry(logs)
e3.place(x=270, y=20, width=30)

e4 = ttk.Entry(logs)
e4.place(x=340, y=20, width=30)
```

```
e5 = ttk.Entry(logs)
e5.place(x=400, y=20, width=30)

# Etiketes

l0 = ttk.Label(logs, text="y = ")
l0.place(x=92, y=20)

l1 = ttk.Label(logs, text="x^4 +")
l1.place(x=155, y=20)

l2 = ttk.Label(logs, text="x^3 +")
l2.place(x=233, y=20)

l3 = ttk.Label(logs, text="x^2 +")
l3.place(x=305, y=20)

l4 = ttk.Label(logs, text="x +")
l4.place(x=377, y=20)

# Obligāta rindiņa, lai logs būtu redzāms visu laiku

logs.mainloop()
```

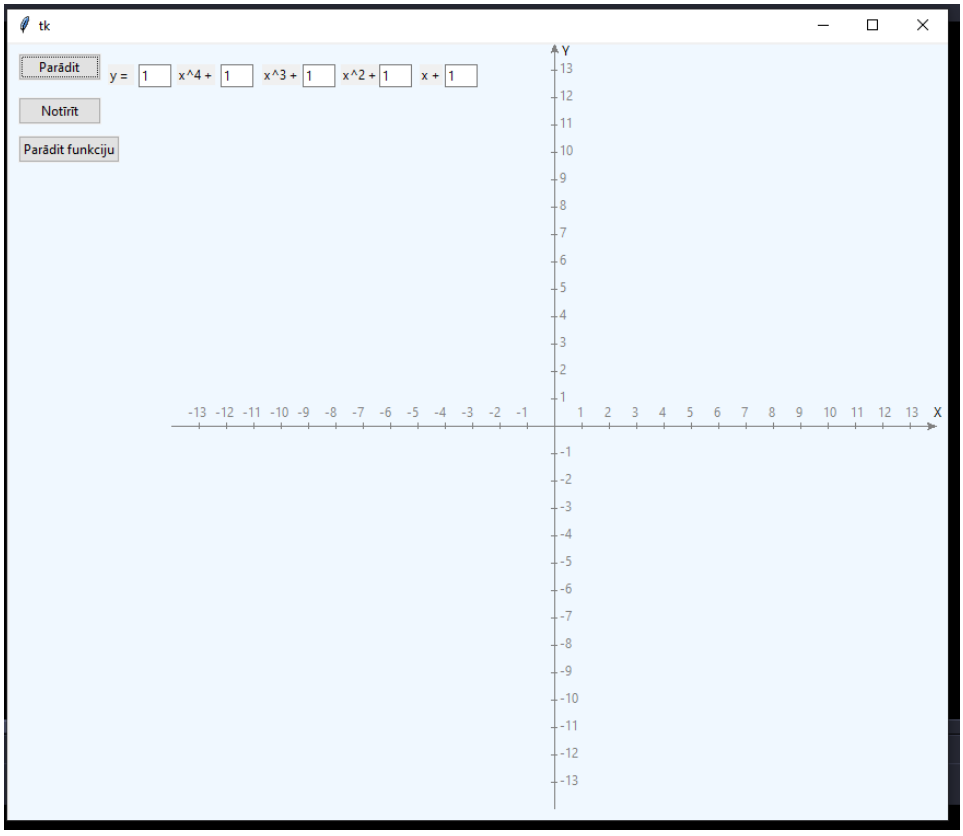
Testa piemēri:

1)

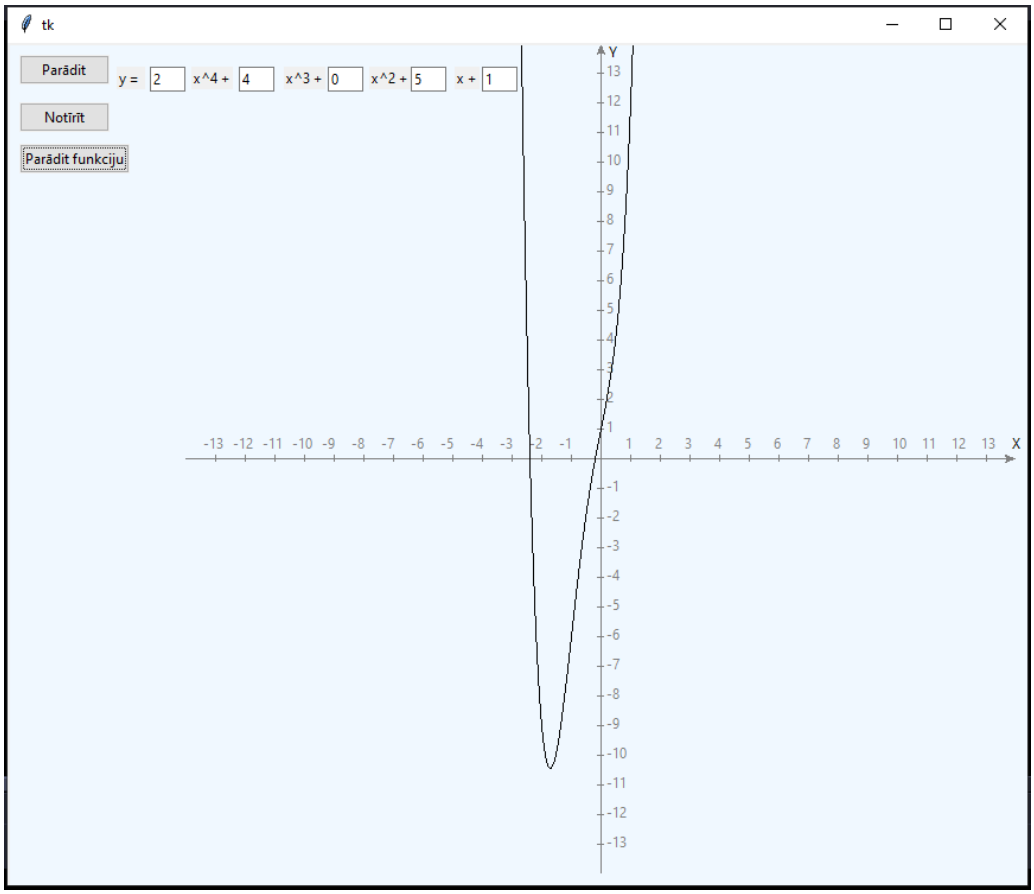


2)

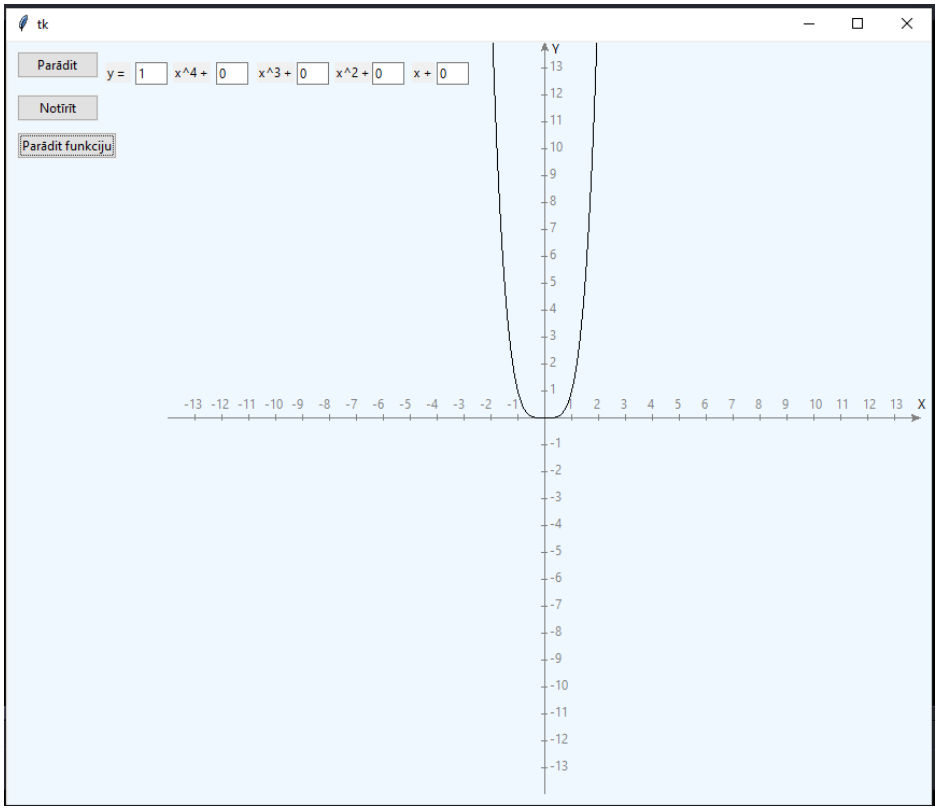
3)


tk
Parādit
y = 1 x^4 + 1 x^3 + 1 x^2 + 1 x + 1
Notīrīt
Parādit funkciju
Y
X

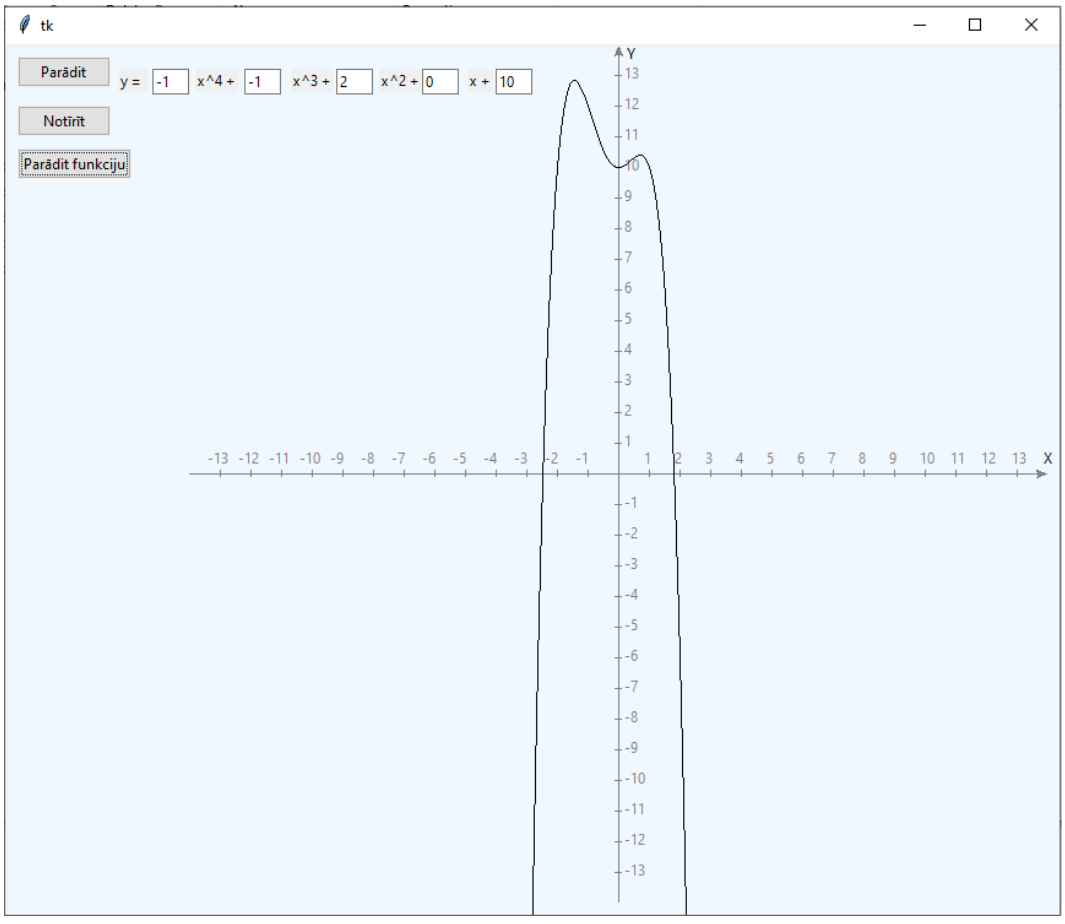
4)


tk
Parādit
y = 2 x^4 + 4 x^3 + 0 x^2 + 5 x + 1
Notīrīt
Parādit funkciju
Y
X

5)


tk
Parādit
y = 1 x^4 + 0 x^3 + 0 x^2 + 0 x + 0
Notīrīt
Parādit funkciju
Y
X

6)


tk
Parādit
y = -1 x^4 + -1 x^3 + 2 x^2 + 0 x + 10
Notīrīt
Parādit funkciju
Y
X

5. uzdevums

Sastādīt programmu, kas izveido “auseklīšu lietu”, t.i., uz ekrāna tiek nodrukāts uz labu laimi izvēlētā skaitā, uz labu laimi izvēlētā vietā, krāsas un izmēra auseklīši. Viena auseklīša uzzīmēšanai jāizveido procedūra, kura “prot” uzzīmēt norādītāja vietā, norādītajā krāsā un izmērā vienu auseklīti.

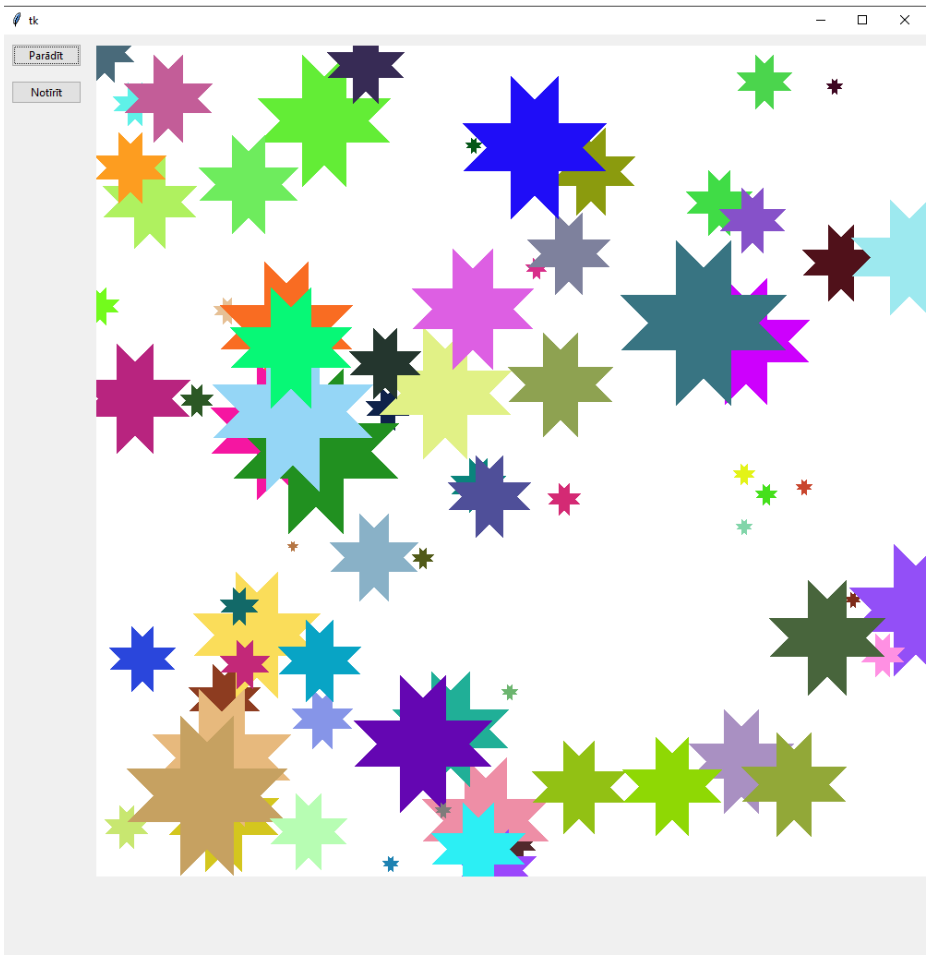
Kods:

```
# Importējam Tkinter un random bibliotēkas

import random

import tkinter

from tkinter import ttk

from random import randrange


# Definējam procedūru, kas uzzīmē ausekli norādītajā vietā, krāsā un izmērā

def draw_auseklis(x, y, r, color):

   #Uz ekrāna uzzīmējam ausekli

   kanva.create_polygon(x,y-2*r,  x+r,y-3*r,  x+r,y-r,  x+3*r,y-r,  x+2*r,y,  x+3*r,y+r,  x+r,y+r,  x+r,y+3*r,  x,y+2*r,  x-r,y+3*r,  x-r,y+r,  x-3*r,y+r,  x-2*r,y,  x-3*r, y-r,  x-r,y-r, x-r,y-3*r, fill=color)


def paradit():

   auseklu_skaits = int(random.randint(1,100)) # Iegūstam nejaušu skaitli no 1 līdz 100, lai noteiktu ausekļu skaitu


   # Iegūstam nejaušas krāsas un izmērus ausekļiem

   for i in range(auseklu_skaits):

      x = int(random.randint(1,900)) # centrs pēc x

      y = int(random.randint(1,900)) # centrs pēc y

      r = int(random.randint(1,30)) # izmers

      color='#%02x%02x%02x' % (randrange(256), randrange(256), randrange(256)) #nejaušas krāsa

      draw_auseklis(x,y, r, color)


def notirit():
```

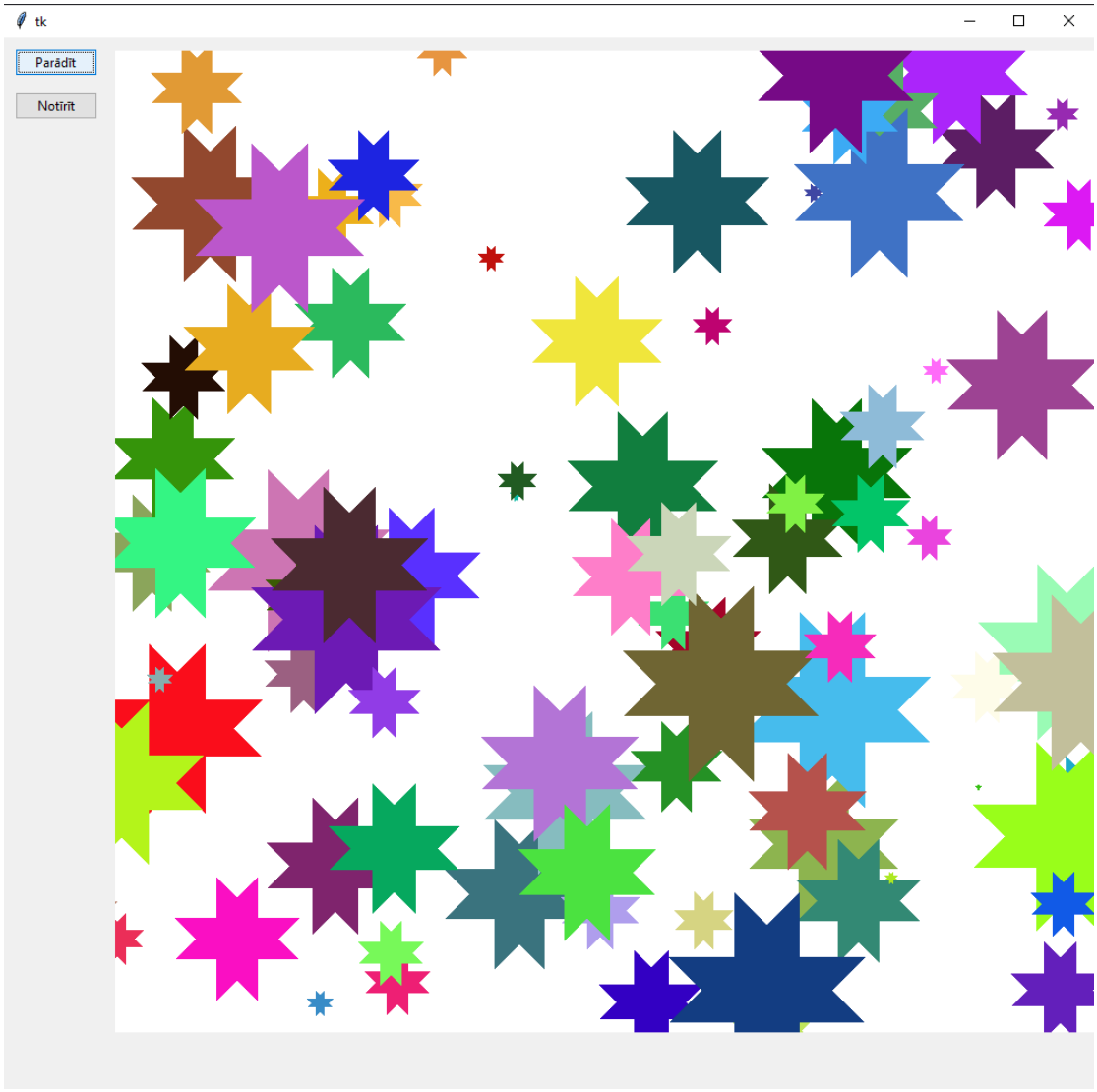
```
    kanva.delete("all") # notirisanai


logs = tkinter.Tk()
logs.geometry("1000x1000")
kanva = tkinter.Canvas(logs, bg="white", height=900, width=900)
kanva.place(x=100, y=10)
poga1= ttk.Button(logs, text="Parādīt", command=paradit)
poga1.place(x=10,y=10)
poga2 = ttk.Button(logs, text="Notīrīt", command=notirit)
poga2.place(x=10, y=50)
logs.mainloop()
```

Testa piemēri:

1)



2)



3)