1. praktiskais darbs

0. uzdevums

Eksāmena uzdevumu kopīga izskatīšana.

1. uzdevums

Sastādīt programmu, kas aprēķina izteiksmes

$$1+\cfrac{1}{2+\cfrac{1}{3+\cfrac{1}{4+\cfrac{1}{5+\cfrac{1}{6+\frac{1}{\dots}}}}}}$$

$$N+\frac{1}{N+1}$$

vērtību, ja N ir naturāls skaitlis, ko lietotājs ievada no tastatūras.

Veikt ievaddatu korektuma pārbaudi.

Kods:

```
# Programmas nosaukums: Noteiktas izteiksmes vērtība

# 1. uzdevums (1MPR01_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas aprēķina noteiktas izteiksmes vērtību, ja N ir naturāls skaitlis,

# ko lietotājs ievada no tastatūras. Veikt ievaddatu korektuma pārbaudi.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0


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 izteiksmes_vertiba(n):
    # Funkcija atrgriež uzdevumā norādītas izteiksmes vērtību.
    # n - naturāls skaitlis
    s = n + 1
    for i in range(n, 0, -1):
        s = i + 1 / s
    return s

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


# Prasam ievādīt naturālo skaitli, ievādam kā simbolu virkni
n = input("Ievadiet naturalo skaitli ==> ")

# Kāmer n nav naturāls skaitlis prasam ievadit to vēlreiz
while is_natural(n) == False:
    n = input("Nav naturals skaitlis.\nIevadiet naturalo skaitli ==> ")

# Ja tas simbolu virkne ir naturāls skaitlis, tad pārveidojam to int formāta
n = int(n)

# Izsaucam funkciju, kas aprēķina izteiksmes vērtību
print("Dotas izteiksmes vērtība ir: " + str(izteiksmes_vertiba(n)))
```

Testa piemēri:

1)

```
Ievadiet naturalo skaitli ==> 1
Dotas izteiksmes vērtība ir: 1.5
```

2)

```
Ievadiet naturalo skaitli ==> 0
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> nulle
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> -2
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> 3
Dotas izteiksmes vērtība ir: 1.4333333333333333
```

3)

```
Ievadiet naturalo skaitli ==> Nulle
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> Viens
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> Divi
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> 23*pi
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> 13.2
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> 0.2
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> 20
Dotas izteiksmes vērtība ir: 1.4331274267223117
```

4)

```
Ievadiet naturalo skaitli ==> 5
Dotas izteiksmes vērtība ir: 1.433127572016461
```

5)

```
Ievadiet naturalo skaitli ==> --1231
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> .12
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==>
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==>
Nav naturals skaitlis.
Ievadiet naturalo skaitli ==> 1000
Dotas izteiksmes vērtība ir: 1.4331274267223117
```

2. uzdevums

Uzrakstīt programmu, kas atrod vienādojuma $\boldsymbol{ax^3 + by^2 + cz + d = 0}$ visus atrisinājumus veselos skaitļos intervālā no -10 līdz 10 (abus galapunktus ieskaitot), koeficientus a, b, c un d ievada no tastatūras.

Kods:

```
# Programmas nosaukums: Vienādojuma ax3+by2+cz+d=0 atrisināšana

# 2. uzdevums (1MPR01_Vladislavs_Babaņins)

# Uzrakstīt programmu, kas atrod vienādojuma ax3+by2+cz+d=0 visus atrisinājumus veselos skaitļos intervālā no -10 līdz 10

# (abus galapunktus ieskaitot), koeficientus a, b, c un d ievada no tastatūras

# Programmas autors: Vladislavs Babaņins

# Versija 1.0


def solve_equation_ax3_plus_by2_plus_cz_plus_d_equals_0(a, b, c, d):

    # Funkcija atrod visus veselos atrisinājumus vienādojumam ax^3 + by^2 + cz + d = 0

    # Funkcija izmanto pilno pārlasi. Tālak funkcija nodot "kortežus" (x, y, z) veidā kā vienu lielo simbolu virkni

    # a - funkcijas parametrs a (ax^3 + ...)

    # b - funkcijas parametrs b (... + by^2 + ...)

    # c - funkcijas parametrs c (... + cz + ...)

    # d - funkcijas parametrs d (... + d = 0)

    sv = ""
```

```
    for x in range(-10, 11):
        for y in range(-10, 11):
            for z in range(-10, 11):
                if a * x * x * x + b * y * y + c * z + d == 0:
                    sv += "(" + str(x) + ", " + str(y) + ", " + str(z) + ")\n"
    return sv


def is_real(n):
    # Pārbauda vai simbolu virkne ir reāls (racionāls) skaitlis vai nav
    # Ja ir reāls skaitlis, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    try:
        n = float(n)
    except:
        return False
    else:
        return True

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


a = input("Ievadiet koeficientu a ==> ")

while is_real(a) == False:
    a = input("Kļūda! a nav reāls skaitlis.\nIevadiet koeficientu a ==> ")

b = input("Ievadiet koeficientu b ==> ")
```

```
while is_real(b) == False:
    b = input("Kļūda! b nav reāls skaitlis.\nIevadiet koeficientu b ==> ")

c = input("Ievadiet koeficientu c ==> ")

while is_real(c) == False:
    c = input("Kļūda! c nav reāls skaitlis.\nAIevadiet koeficientu c ==> ")

d = input("Ievadiet koeficientu d ==> ")

while is_real(d) == False:
    d = input("Kļūda! d nav reāls skaitlis.\nIevadiet koeficientu d ==> ")

a = float(a)
b = float(b)
c = float(c)
d = float(d)

sv = solve_equation_ax3_plus_by2_plus_cz_plus_d_equals_0(a, b, c, d)

if sv == "":
    print("\nVienādojumam " + str(a) + "*x^3 + " + str(b) + "*y^2 + " + str(c) + "*z + " + str(d) + " = 0 nav atrisinājumus veseļos skaitļos intervāla [-10;10]")
else:
    print("\nVienādojuma " + str(a) + "*x^3 + " + str(b) + "*y^2 + " + str(c) + "*z + " + str(d) + " = 0 atrisinājumi veseļos skaitļos intervāla [-10;10] ir sekojoši skaitļu korteži (x,y,z): \n")

# Izvadām risinājuma "kortežus" (x, y, z) veidā.
print(sv)
```

Testa piemēri:

1)

```
Ievadiet koeficientu a ==> 1
Ievadiet koeficientu b ==> 2
Ievadiet koeficientu c ==> 3
Ievadiet koeficientu d ==> 4

Vienādojuma 1.0*x^3 + 2.0*y^2 + 3.0*z + 4.0 = 0 atrisinājumi veseļos skaitļos intervāla [-10;10] ir sekojoši skaitļu korteži (x,y,z):

(-6, -10, 4)
(-6, 10, 4)
(-4, -6, -4)
(-4, 6, -4)
(-3, -5, -9)
(-3, -4, -3)
(-3, -2, 5)
(-3, -1, 7)
(-3, 1, 7)
(-3, 2, 5)
(-3, 4, -3)
(-3, 5, -9)
(-1, -3, -7)
(-1, 0, -1)
(-1, 3, -7)
(0, -2, -4)
(0, -1, -2)
(0, 1, -2)
(0, 2, -4)
(2, -3, -10)
(2, 0, -4)
(2, 3, -10)
```

2)

```
Ievadiet koeficientu a ==> 2
Ievadiet koeficientu b ==> 3
Ievadiet koeficientu c ==> 4
Ievadiet koeficientu d ==> 5

Vienādojuma 2.0*x^3 + 3.0*y^2 + 4.0*z + 5.0 = 0 atrisinājumi veseļos skaitļos intervāla [-10;10] ir sekojoši skaitļu korteži (x,y,z):

(-4, -7, -6)
(-4, 7, -6)
(-2, -3, -4)
(-2, -1, 2)
(-2, 1, 2)
(-2, 3, -4)
(0, -3, -8)
(0, -1, -2)
(0, 1, -2)
(0, 3, -8)
(2, -1, -6)
(2, 1, -6)
```

3)

```
Ievadiet koeficientu a ==> 5
Ievadiet koeficientu b ==> 6
Ievadiet koeficientu c ==> 7
Ievadiet koeficientu d ==> 8

Vienādojuma 5.0*x^3 + 6.0*y^2 + 7.0*z + 8.0 = 0 atrisinājumi veseļos skaitļos intervāla [-10;10] ir sekojoši skaitļu korteži (x,y,z):

(0, -1, -2)
(0, 1, -2)
```

4)

```
Ievadiet koeficientu a ==> 8
Ievadiet koeficientu b ==> 4984
Ievadiet koeficientu c ==> 12
Ievadiet koeficientu d ==> 1

Vienādojumam 8.0*x^3 + 4984.0*y^2 + 12.0*z + 1.0 = 0 nav atrisinājumus veseļos skaitļos intervāla [-10;10]
```

5)

```
Ievadiet koeficientu a ==> Nulle
Kļūda! a nav reāls skaitlis.
Ievadiet koeficientu a ==> Divi
Kļūda! a nav reāls skaitlis.
Ievadiet koeficientu a ==> Tris
Kļūda! a nav reāls skaitlis.
Ievadiet koeficientu a ==>
Kļūda! a nav reāls skaitlis.
Ievadiet koeficientu a ==> 1
Ievadiet koeficientu b ==> Viens
Kļūda! b nav reāls skaitlis.
Ievadiet koeficientu b ==> 1
Ievadiet koeficientu c ==> 2
Ievadiet koeficientu d ==> 3

Vienādojuma 1.0*x^3 + 1.0*y^2 + 2.0*z + 3.0 = 0 atrisinājumi veseļos skaitļos intervāla [-10;10] ir sekojoši skaitļu korteži (x,y,z):

(-4, -9, -10)
(-4, -7, 6)
(-4, 7, 6)
(-4, 9, -10)
(-3, -6, -6)
(-3, -4, 4)
(-3, -2, 10)
(-3, 2, 10)
(-3, 4, 4)
(-3, 6, -6)
(-2, -5, -10)
(-2, -3, -2)
(-2, -1, 2)
(-2, 1, 2)
(-2, 3, -2)
(-2, 5, -10)
(-1, -4, -9)
(-1, -2, -3)
(-1, 0, -1)
(-1, 2, -3)
(-1, 4, -9)
(0, -3, -6)
(0, -1, -2)
(0, 1, -2)
(0, 3, -6)
(1, -4, -10)
(1, -2, -4)
(1, 0, -2)
(1, 2, -4)
(1, 4, -10)
(2, -3, -10)
(2, -1, -6)
(2, 1, -6)
(2, 3, -10)
```

3. uzdevums

Sastādīt programmu, kas aprēķina izteismes

$$\arcsin(x) = x + \frac{1}{2}\cdot\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\cdot\frac{x^5}{5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot\frac{x^7}{7} + \frac{1\cdot 3\cdot 5\cdot 7}{2\cdot 4\cdot 6\cdot 8}\cdot\frac{x^9}{9} + \cdots$$

Vērtību ar precizitāti 10^{-6}, ja zināms, ka $|x| < 1$ un to lietotājs ievada no tastatūras, pieņemot, ka izteksmes precizitāti nosaka pēdējais saskaitāmais. Pārbaudīt ievades datu korektumu!

Kods:

```
# Programmas nosaukums: Noteiktas izteiksmes vērtība

# 3. uzdevums (1MPR01_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas aprēķina izteiksmes arcsin(x) vērtību ar precizitāti 10^(-6), ka zināms, ka

# ja zināms, ka abs(x) < 1 un to lietotājs ievada no tastatūras, pieņemot, ka izteiksmes precizitāti nosaka pēdējais saskaitāmais.

# Pārbaudīt ievades datu korektumu
```

```
# Programmas autors: Vladislavs Babaņins
# Versija 1.0

import math



def arcsin(x, PR):
    # Atgriež arcisn(x) vērtību
    # x - arcins(x) - funkcijas arguments
    # PR - precizitāte (nosaka pedējais saskaitamais)
    n = 0
    s = x
    y = x
    while abs(y / (n + 1)) > PR:
        n = n + 2
        y = y * x * x * (n - 1) / n
        s += y / (n + 1)
    return s



def is_real_and_abs_mazaks_neka_viens(n):
    # Pārbauda vai simbolu virkne n ir reāls skaitls un abs(n) < 1
    # Atgriež True, ja izpildas abi nosacījumi.
    # Atgriež "Nav reals skaitlis", ja simbolu virkne n nav reāls skaitlis
    # Atgriež "Ir reals skaitlis, bet abs(n) >= 1", ja ja simbolu virkne ir reāls skaitlis, bet abs(n) >= 1
    # n - simbolu virkne
    try:
        n = float(n)
    except:
        return "Nav reals skaitlis"
    else:
```

```
        n = float(n)
        if abs(n) >= 1:
            return "Ir reals skaitlis, bet abs(n) >= 1"
        else:
            return True


def is_real(n):
    # Pārbauda vai simbolu virkne ir reāls (racionāls) skaitlis vai nav
    # Ja ir reāls skaitlis, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    try:
        n = float(n)
    except:
        return False
    else:
        return True

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


x = input("Ievadi funkcijas argumentu x ===> ")


while is_real_and_abs_mazaks_neka_viens(x) == "Nav reals skaitlis" or
is_real_and_abs_mazaks_neka_viens(x) == "Ir reals skaitlis, bet abs(n) >= 1":
    if is_real_and_abs_mazaks_neka_viens(x) == "Nav reals skaitlis":
        x = input("Kļūda! Nav reāls skaitlis\nIevadiet reālo skaitli ==> ")
    else:
        x = input("Kļūda! abs(x) >= 1\nIevadiet tādu reālo skaitli x, lai abs(x) < 1 ==> ")
```

```
PR = 0.000001

x = float(x)

result = arcsin(x, PR)

round_result = round(result, 6)

print("arcsin(" + str(x) + ") = " + str(round_result) + " radiāni, ar precizitāti " + str(PR))

print("arcsin(" + str(x) + ") = " + str(round(round_result / math.pi * 180, 6)) + " grādi, ar precizitāti " + str(PR))
```

Testa piemēri:

1)

```
Ievadi funkcijas argumentu x ===> 1
Kļūda! abs(x) >= 1
Ievadiet tādu reālo skaitli x, lai abs(x) < 1 ==> nulle koma pieci
Kļūda! Nav reāls skaitlis
Ievadiet reālo skaitli ==> 0.5
arcsin(0.5) = 0.523599 ar precizitāti 1e-06
```

2)

```
Ievadi funkcijas argumentu x ===> liels arguments
Kļūda! Nav reāls skaitlis
Ievadiet reālo skaitli ==>
Kļūda! Nav reāls skaitlis
Ievadiet reālo skaitli ==> -2
Kļūda! abs(x) >= 1
Ievadiet tādu reālo skaitli x, lai abs(x) < 1 ==> -15
Kļūda! abs(x) >= 1
Ievadiet tādu reālo skaitli x, lai abs(x) < 1 ==> -.5
arcsin(-0.5) = -0.523599 ar precizitāti 1e-06
```

3)

```
Ievadi funkcijas argumentu x ===> 0.9999
arcsin(0.9999) = 1.556654 ar precizitāti 1e-06
```

4)

```
Ievadi funkcijas argumentu x ===> -0.0000000001
arcsin(-1e-10) = -0.0 ar precizitāti 1e-06
```

5)

```
Ievadi funkcijas argumentu x ===> 0
arcsin(0.0) = 0.0 ar precizitāti 1e-06
```

6)

```
Ievadi funkcijas argumentu x ===> 10000
Kļūda! abs(x) >= 1
Ievadiet tādu reālo skaitli x, lai abs(x) < 1 ==> -10000
Kļūda! abs(x) >= 1
Ievadiet tādu reālo skaitli x, lai abs(x) < 1 ==> 0.0002
arcsin(0.0002) = 0.0002 ar precizitāti 1e-06
```

4. uzdevums

Sastādīt programmu, kas noskaidro, no kādiem pirmskaitļiem var izveidot četrus lietotāja patvaļīgi ievadītos naturālos skaitļus (skaitļus veido tikai kā pirmskaitļu reizinājumu, piemēram, 100=2*2*5*5). Pirmskaitļi ir jāpaziņo augošā secībā.

Piemēram:

Ievadiet pirmo skaitli => 50

Ievadiet otro skaitli => 75

Ievadiet trešo skaitli => 100

Ievadiet ceturto skaitli => 125

Ievadītos skaitļus var izveidot no šādiem pirmskaitļiem: ***2, 3, 5.***

Kods:

```
# Programmas nosaukums: Noteiktas izteiksmes vērtība

# 4. uzdevums (1MPR01_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas noskaidro, no kādiem pirmskaitļiem var izveidot četrus lietotāja

# patvaļīgi ievadītos naturālos skaitļus (skaitļus veido tikai kā pirmskaitļu reizinājumu,

# piemēram, 100=2*2*5*5). Pirmskaitļi ir jāpaziņo augošā secībā.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0


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 sadalisana_uz_pirmskaitliem_kuri_neatkartojas(x):
    # Sadala skaitli x uz pirmskaitliem, kuri neatkartojas
    # Ja ir naturāls skaitlis, tad True. Ja nav tad False.
    # x - naturāls skaitlis
    n = 2
    r = ""
    while x > 1:
        if x % n == 0:
            r = r + str(n) + " "
            x = x // n
            while x % n == 0:
                x = x // n
        n += 1

    if r == "":
        return 1
    return r

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


a = input("Ievadiet pirmo skaitli ==> ")
```

```
while is_natural(a) == False:
    a = input("Kļūda! Pirmais skaitlis nav reāls skaitlis.\nIevadiet pirmo skaitli ==> ")

b = input("Ievadiet otro skaitli ==> ")
while is_natural(b) == False:
    b = input("Kļūda! Otrais skaitlis nav reāls skaitlis.\nIevadiet otro skaitli ==> ")

c = input("Ievadiet trešo skaitli ==> ")
while is_natural(c) == False:
    c = input("Kļūda! Trešais skaitlis nav reāls skaitlis.\nIevadiet trešo skaitli ==> ")

d = input("Ievadiet ceturto skaitli ==> ")
while is_natural(d) == False:
    d = input("Kļūda! Ceturtais skaitlis nav reāls skaitlis.\nIevadiet ceturto skaitli ==> ")

a = int(a)
b = int(b)
c = int(c)
d = int(d)

x = a * b * c * d

result = sadalisana_uz_pirmskaitliem_kuri_neatkartojas(x)

print("Ievadītos skaitļus var izveidot no šādiem pirmskaitļiem: " + str(result))
```

Testa piemēri:

1)

```
Ievadiet pirmo skaitli ==> 1
Ievadiet otro skaitli ==> 2
Ievadiet trešo skaitli ==> 3
Ievadiet ceturto skaitli ==> 4
Ievadītos skaitļus var izveidot no šādiem pirmskaitļiem: 2 3
```

2)

```
Ievadiet pirmo skaitli ==> 2
Ievadiet otro skaitli ==> 3
Ievadiet trešo skaitli ==> 4
Ievadiet ceturto skaitli ==> 5.5
Kļūda! Ceturtais skaitlis nav reāls skaitlis.
Ievadiet ceturto skaitli ==> faf
Kļūda! Ceturtais skaitlis nav reāls skaitlis.
Ievadiet ceturto skaitli ==> 7.6
Kļūda! Ceturtais skaitlis nav reāls skaitlis.
Ievadiet ceturto skaitli ==> 5
Ievadītos skaitļus var izveidot no šādiem pirmskaitļiem: 2 3 5
```

3)

```
Ievadiet pirmo skaitli ==> 512
Ievadiet otro skaitli ==> 1337
Ievadiet trešo skaitli ==> 22
Ievadiet ceturto skaitli ==> 5661
Ievadītos skaitļus var izveidot no šādiem pirmskaitļiem: 2 3 7 11 17 37 191
```

4)

```
Ievadiet pirmo skaitli ==> 1
Ievadiet otro skaitli ==> 2
Ievadiet trešo skaitli ==> 3
Ievadiet ceturto skaitli ==> 4
Ievadītos skaitļus var izveidot no šādiem pirmskaitļiem: 2 3
```

5)

```
Ievadiet pirmo skaitli ==> 12
Ievadiet otro skaitli ==> 13
Ievadiet trešo skaitli ==> 14
Ievadiet ceturto skaitli ==> 15
Ievadītos skaitļus var izveidot no šādiem pirmskaitļiem: 2 3 5 7 13
```

5. uzdevums

Sastādīt programmu, kas uzzīmē kardioīdu, kuras vienādojums parametriskā formā ir

$$\begin{cases} x = a(2cos\varphi - cos2\varphi) \\ y = a(2sin\varphi - sin2\varphi) \end{cases}$$

Kur $\varphi \in [0; 2\pi]$, bet parametru a ievada no tastatūras. Koordinātu sistēma jāizvēlas tā, lai attēls būtu pa visu ekrānu, neatkarīgi no a vērtības.

Kods:

```
# Programmas nosaukums: Kardīoda zīmējums

# 5. uzdevums (1MPR01_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas uzzīmē kardioīdu, kuras vienādojums parametriskā formā ir

# x=a(2cosϕ-cos2ϕ)

# y=a(2sinϕ-sin2ϕ)

# Kur ϕ∈[0;2π], bet parametru a ievada no tastatūras. Koordinātu sistēma jāizvēlas tā, lai attēls būtu pa visu ekrānu, neatkarīgi no a vērtības.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import math

import tkinter

from tkinter import ttk

from tkinter import *



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

    # Aktive pogu "Linija", kura parāda kardioīdu

    poga3['state'] = ACTIVE



def disable_button_linija(): # Dizaktīve pogu "Parādīt funkciju"
```

```
    # Padara pogu "Linija" par neaktīvu, kura parāda kardioīdu
    poga3['state'] = DISABLED


def enable_button_plakne(): # Dizaktīve pogu "Parādīt funkciju"
    # Padara pogu "Plakne" par aktīvu, kura parāda plakni
    poga1['state'] = ACTIVE


def disable_button_plakne(): # Dizaktīve pogu "Parādīt funkciju"
    # # Padara pogu "Plakne" par neaktīvu, kura parāda kardioīdu
    poga1['state'] = DISABLED


def notirit():
    # Kanvas satura dzēšanai un dizaktivē pogu "Līnija"
    kanva.delete("all")
    disable_button_linija()


def is_real_number_in_entry(event):
    # Parbauda vai e1 entry logā ir ierakstīts reāls skaitlis
    # Padara tā, ka poga "Plakne" ir aktīva tikai tad, ja entry logā ir ierakstīts reāls skaitlis
    # Padara tā, ka poga "Līnija" ir aktīva tikai tad, ja tika nospiests uz pogu "Plakne"
    # Tas ir izdārīts, lai nevarētu uzzīmēt līnju, bez uzzimētas plaknes
    # event - simbolu ierakstīšana entry logā
    disable_button_plakne()
    disable_button_linija()
    kanva.delete("all")
    try:
```

```
            float(e1.get()) # Pārbaude no e1 ņemsim simbolu virkni un pārbaudīsim vai to var pārveidot float
        except:
            disable_button_plakne()
            disable_button_linija()
            kanva.delete("all")

        else:
            if float(e1.get()) > 0:
                enable_button_plakne()
            else:
                disable_button_plakne()
                disable_button_linija()
                kanva.delete("all")


    def paradit():
        # Koordinātu plaknes uzzīmēšana, kura pielāgojas atkarība no a (Lai tāda veidā parādīt kardioīda izmēru).
        enable_button_linija()
        a = float(e1.get())
        kanva.create_line(150, 350, 850, 350, fill="gray")
        kanva.create_line(845, 345, 850, 350, fill="gray")
        kanva.create_line(845, 355, 850, 350, fill="gray")
        kanva.create_text(845, 330, text="X", anchor="nw", font=("Helvetica", 10), fill="gray")
        kanva.create_line(500, 0, 500, 700, fill="gray")
        kanva.create_line(505, 5, 500, 0, fill="gray")
        kanva.create_line(495, 5, 500, 0, fill="gray")
        kanva.create_text(505, 0, text="Y", anchor="nw", font=("Helvetica", 10), fill="gray")

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

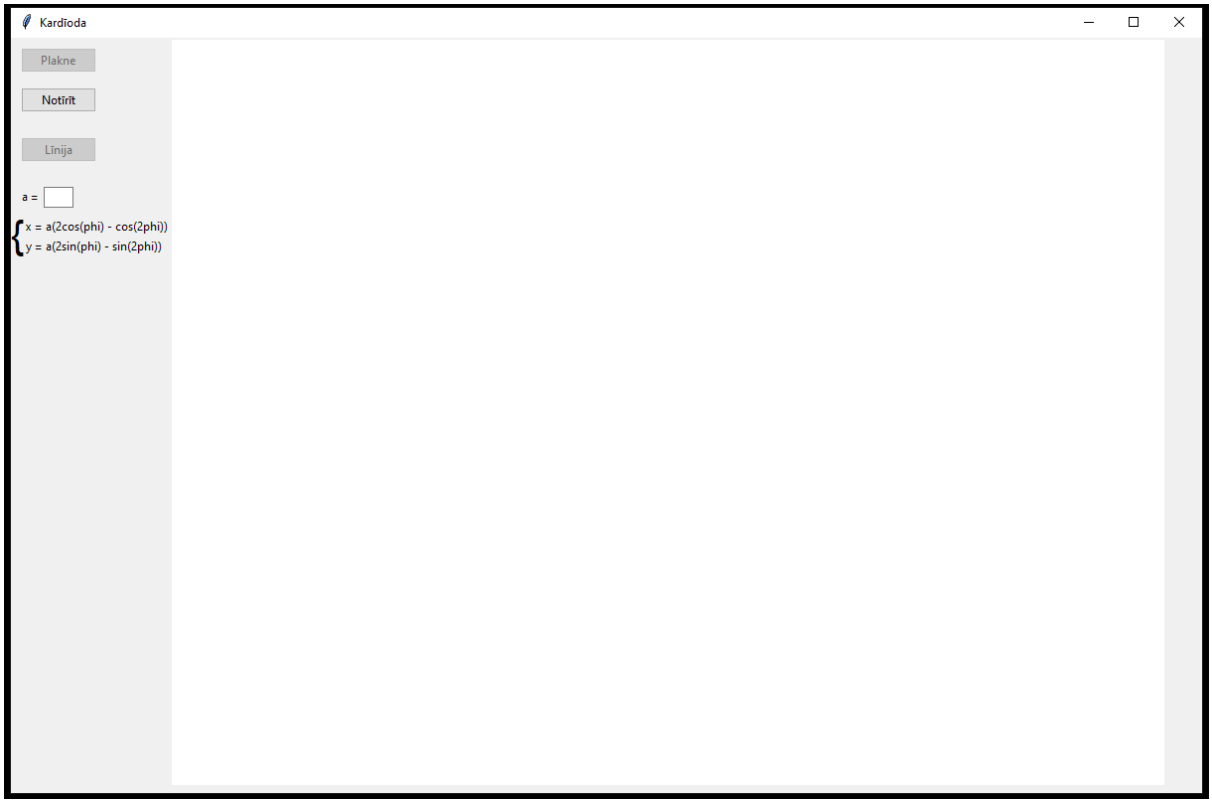
```
        #kanva.create_text(175, 330, text = str(-3*a), anchor = "nw", font = ("Helvetica",10), fill =
"gray")
        #kanva.create_text(591, 330, text = str(a), anchor = "nw", font=("Helvetica", 10), fill =
"gray")

        for i in range(25, 676, 25):
            kanva.create_line(497, i, 503, i)
        #kanva.create_text(505, 603, text = str(-2.5*a), anchor = "nw", font = ("Helvetica", 10), fill =
"gray")
        #kanva.create_text(505, 80, text = str(2.5*a), anchor = "nw", font = ("Helvetica", 10), fill =
"gray")

        for i in range(-13, 0): # minusiem uz X
            kanva.create_text(490 + i * 25, 330, text=str(i * a / (4)), anchor="nw", fill="gray")

        for i in range(1, 14): # plusiem uz X
            kanva.create_text(496 + i * 25, 330, text=str(i * a / (4)), anchor="nw", fill="gray")

        for i in range(-13, 0): # minusiem uz Y
            kanva.create_text(505, 341 - i * 25, text=str(i * a / (4)), anchor="nw", fill="gray")

        for i in range(1, 14): # minusiem uz X
            kanva.create_text(505, 341 - i * 25, text=str(i * a / (4)), anchor="nw", fill="gray")


    def polari():
        # Līnijas uzzīmēšana

        x0 = 500
        y0 = 350
        a = float(e1.get())
```

```
    lenght = a
    b = 4
    a = b

    b = 2 * x0 + a / 25
    x1 = x0 + a / 25
    y1 = y0

    # Līnija
    for i in range(1, 3600, 1):

        f = i / 1800 * math.pi
        x = a * (2 * math.cos(f) - math.cos(2 * f))

        y = a * (2 * math.sin(f) - math.sin(2 * f))
        y2 = -y * 25 + y0
        x2 = x * 25 + x0

        kanva.create_line(x1, y1, x2, y2)
        x1 = x2
        y1 = y2

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


logs = tkinter.Tk()
logs.title("Kardīoda")
logs.geometry("1200x760")
kanva = tkinter.Canvas(logs, bg="white", height=750, width=1000)
```

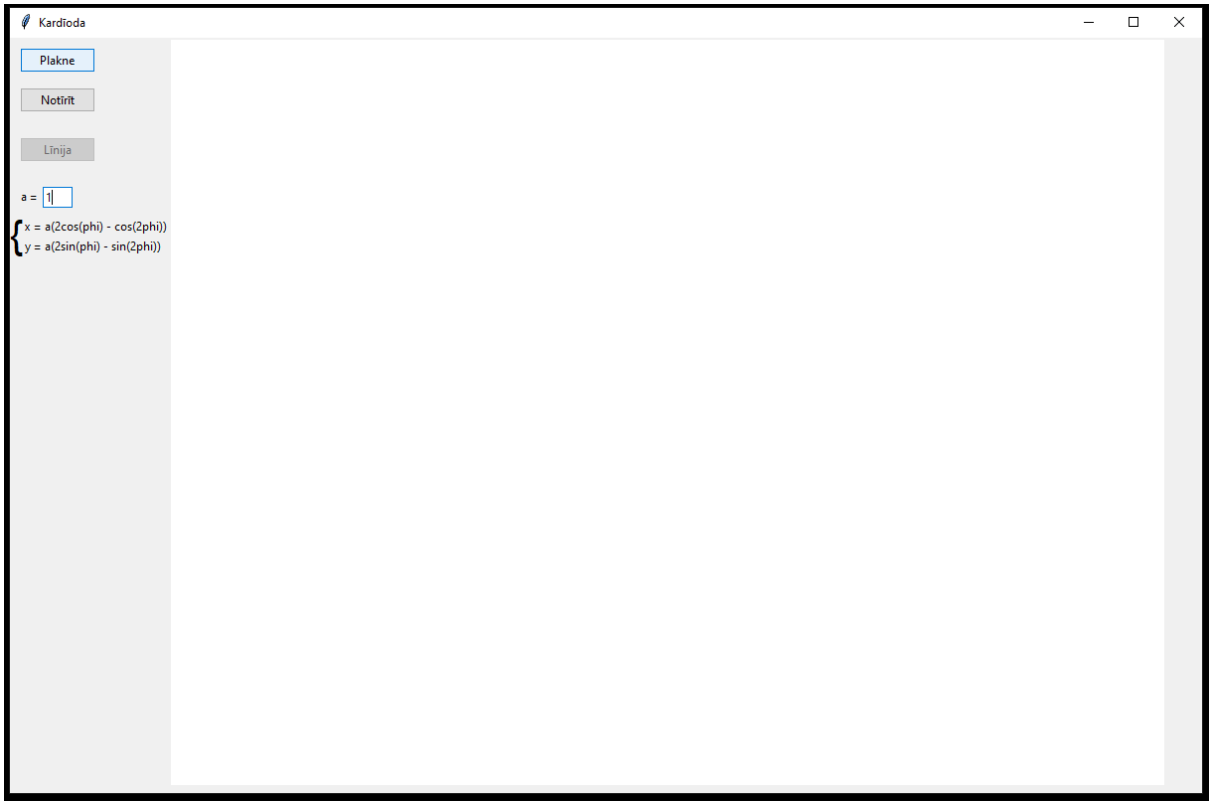
```
        kanva.place(x=160, y=0)
        poga1 = ttk.Button(logs, text="Plakne", command=paradit, state='disabled')
        poga1.place(x=10, y=10)
        poga2 = ttk.Button(logs, text="Notīrīt", command=notirit)
        poga2.place(x=10, y=50)
        poga3 = ttk.Button(logs, text="Līnija", command=polari, state='disabled')
        poga3.place(x=10, y=100)

        # Entry
        e1 = ttk.Entry(logs)
        e1.bind("<KeyRelease>", is_real_number_in_entry) # Izsaucam funkciju
is_real_number_in_entry
        e1.place(x=33, y=150, width=30)

        l0 = ttk.Label(logs, text="x = a(2cos(phi) - cos(2phi))")
        l0.place(x=13, y=180)

        l1 = ttk.Label(logs, text="y = a(2sin(phi) - sin(2phi))")
        l1.place(x=13, y=200)

        l2 = ttk.Label(logs, text="{", font=("Helvetica", 30))
        l2.place(x=-2, y=174)

        l3 = ttk.Label(logs, text="a = ")
        l3.place(x=9, y=150)

        disable_button_linija()

        logs.mainloop()
```

Testa piemēri:

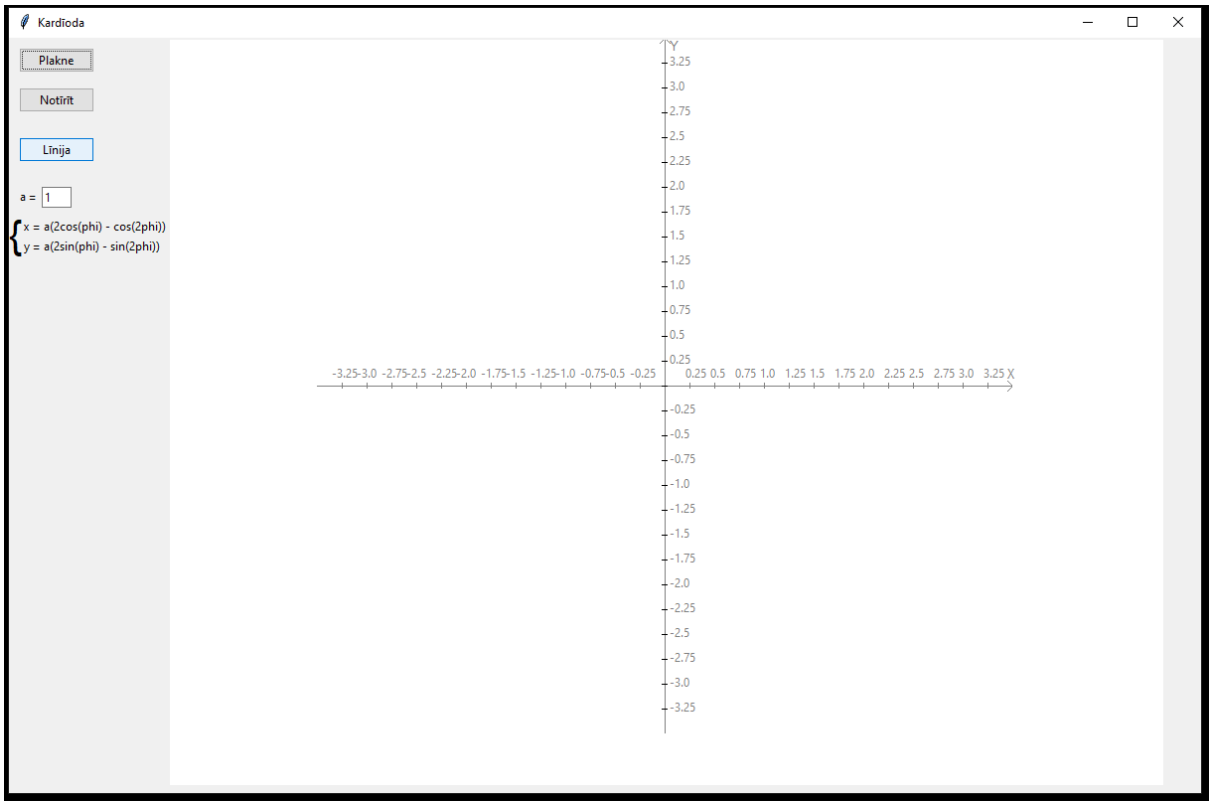
1)



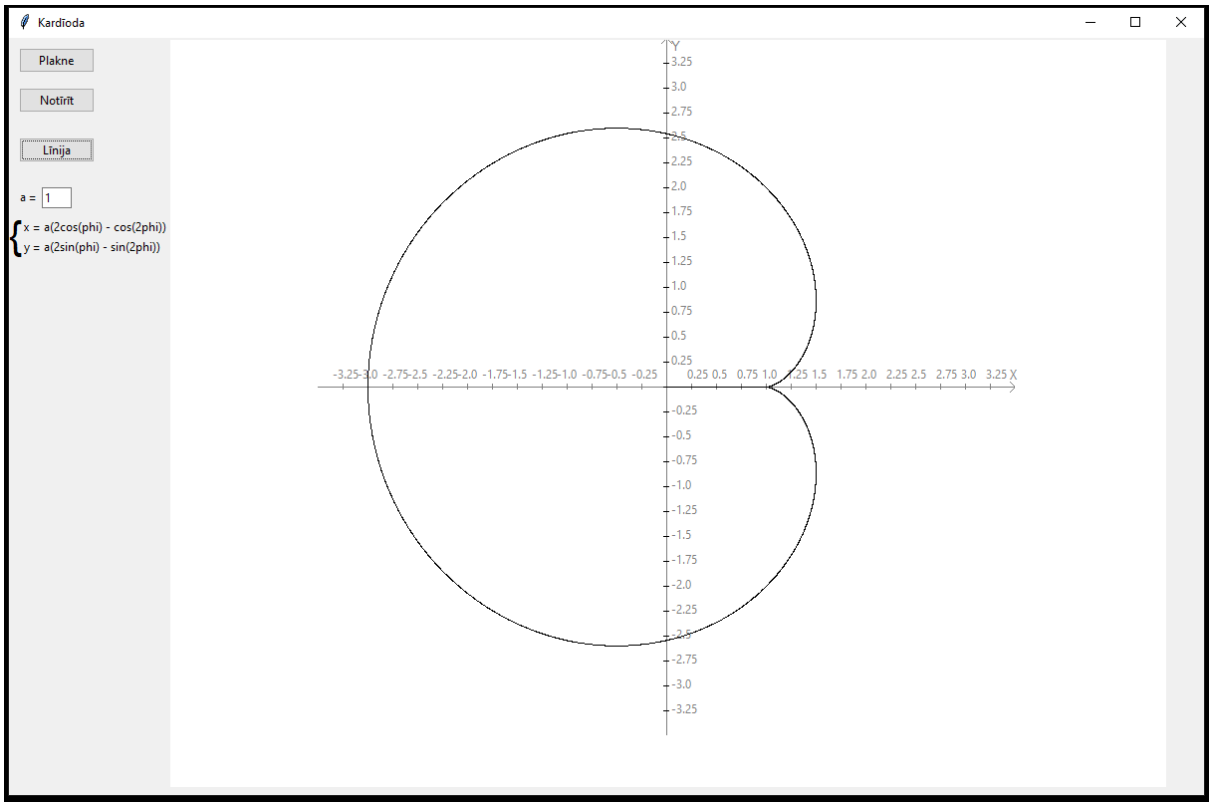
2)



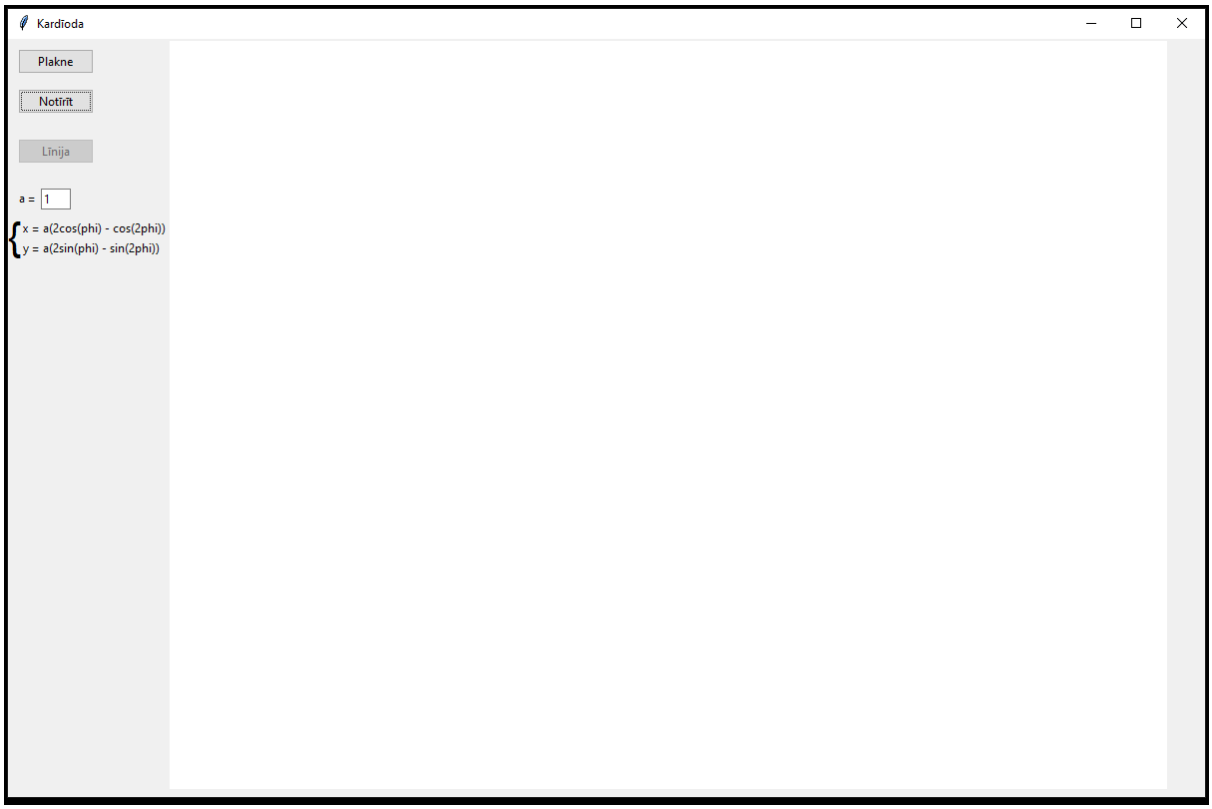
3)


Kardīoda
Plakne
Notīrīt
Līnija
a = 1
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))
Y
X

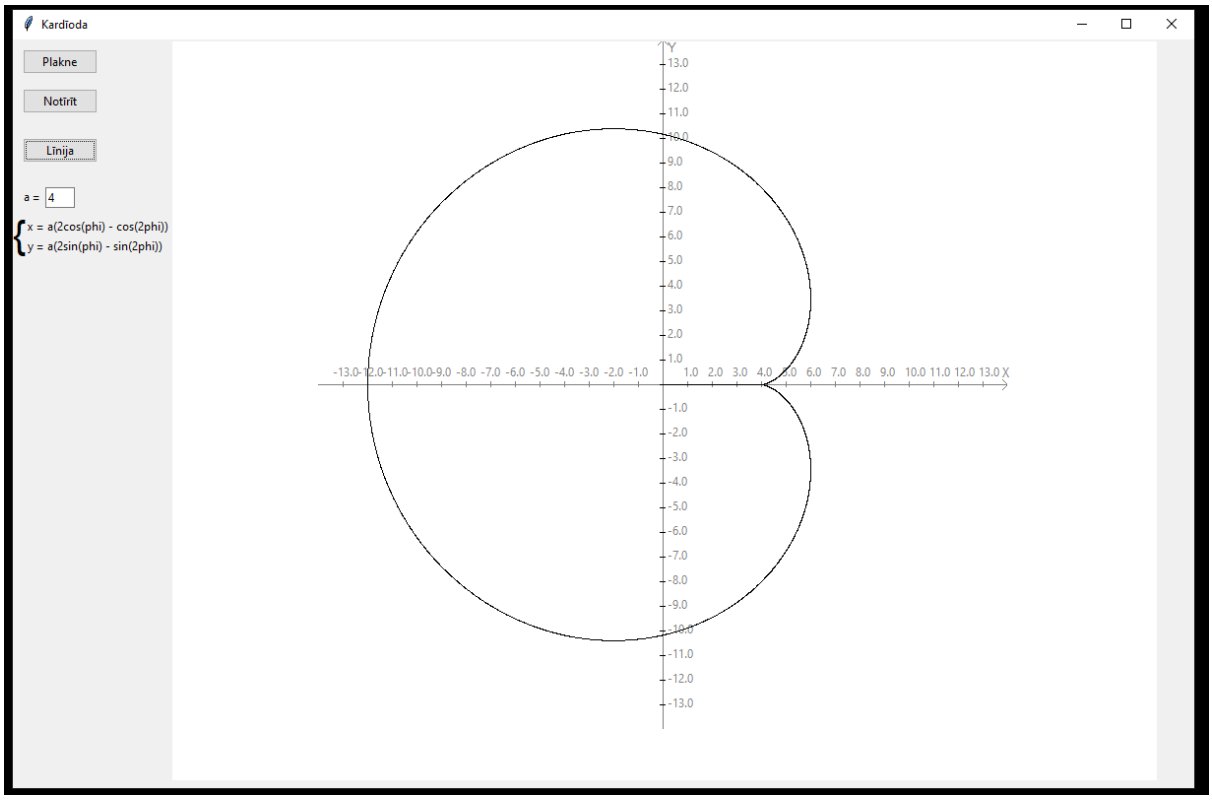
4)


Kardīoda
Plakne
Notīrīt
Līnija
a = 1
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))
Y
X

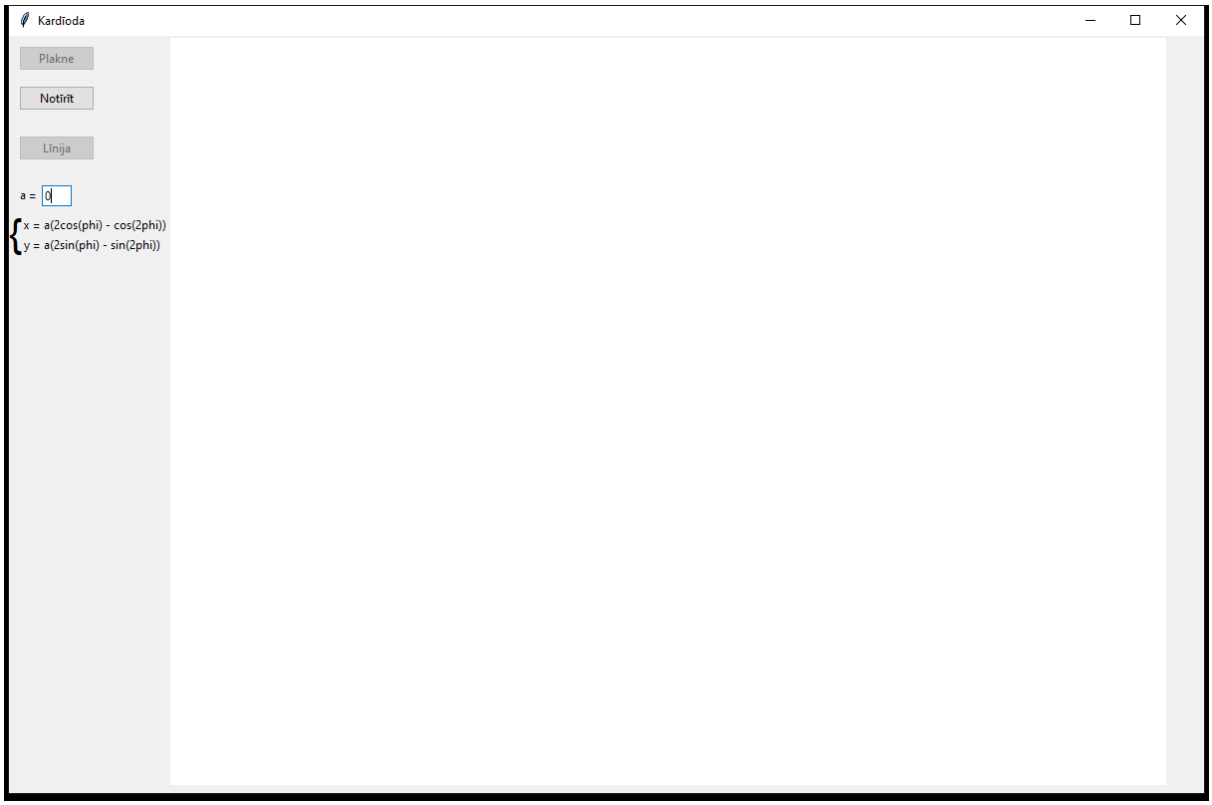
5)


Kardīoda
Plakne
Notīrīt
Līnija
a = 1
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))

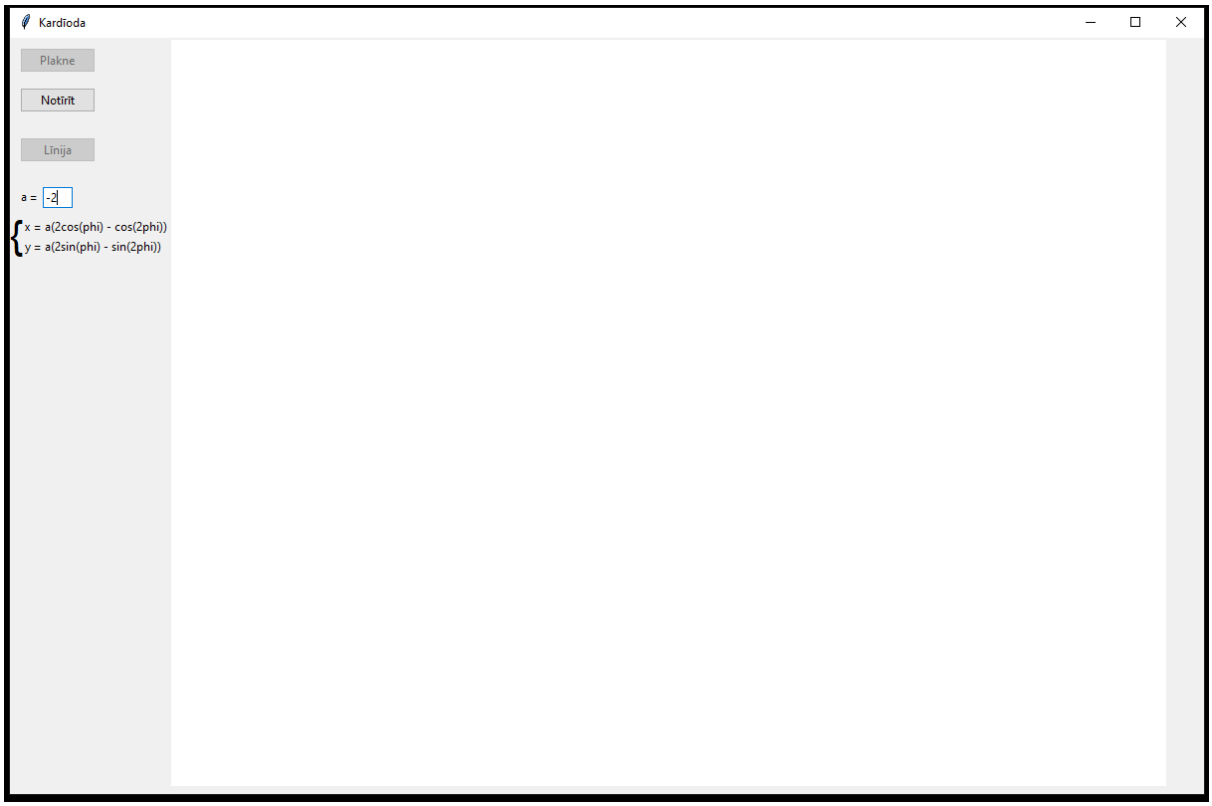
6)


Kardīoda
Plakne
Notīrīt
Līnija
a = 4
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))
Y
X

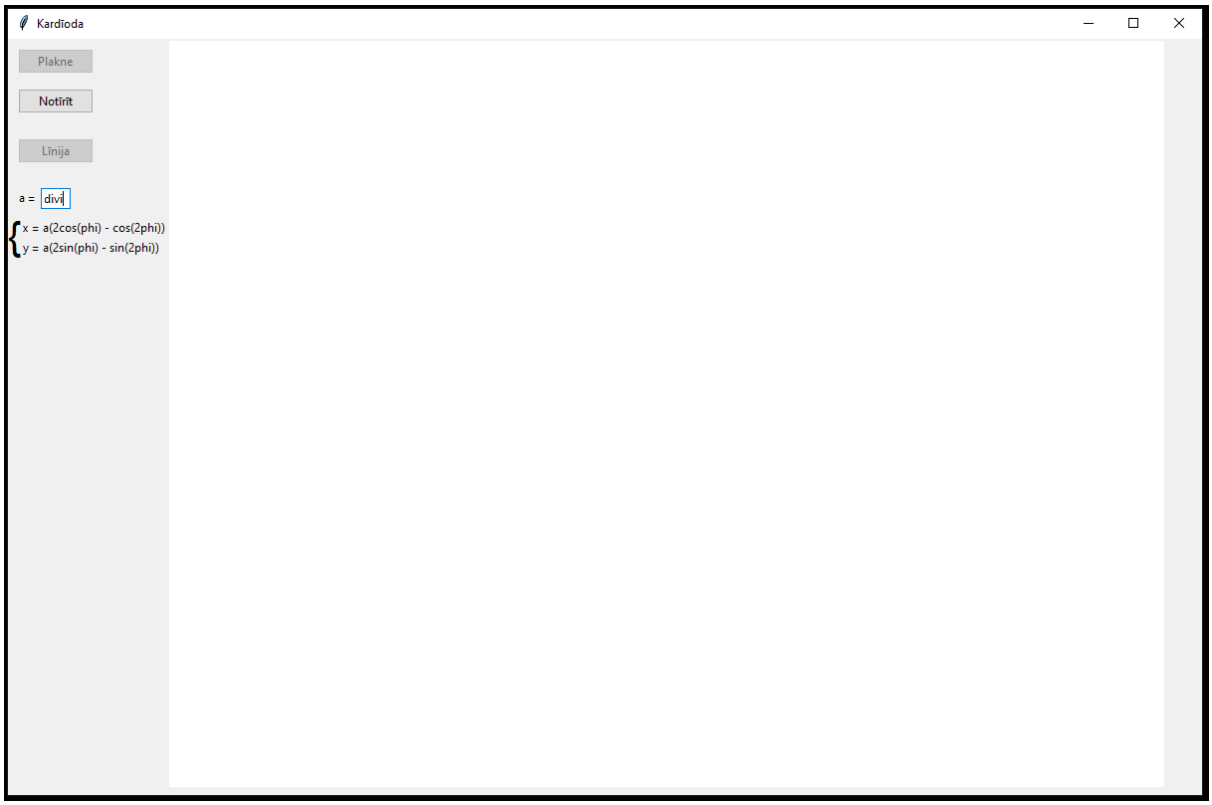
7)


Kardīoda
Plakne
Notīrīt
Līnija
a = 0
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))

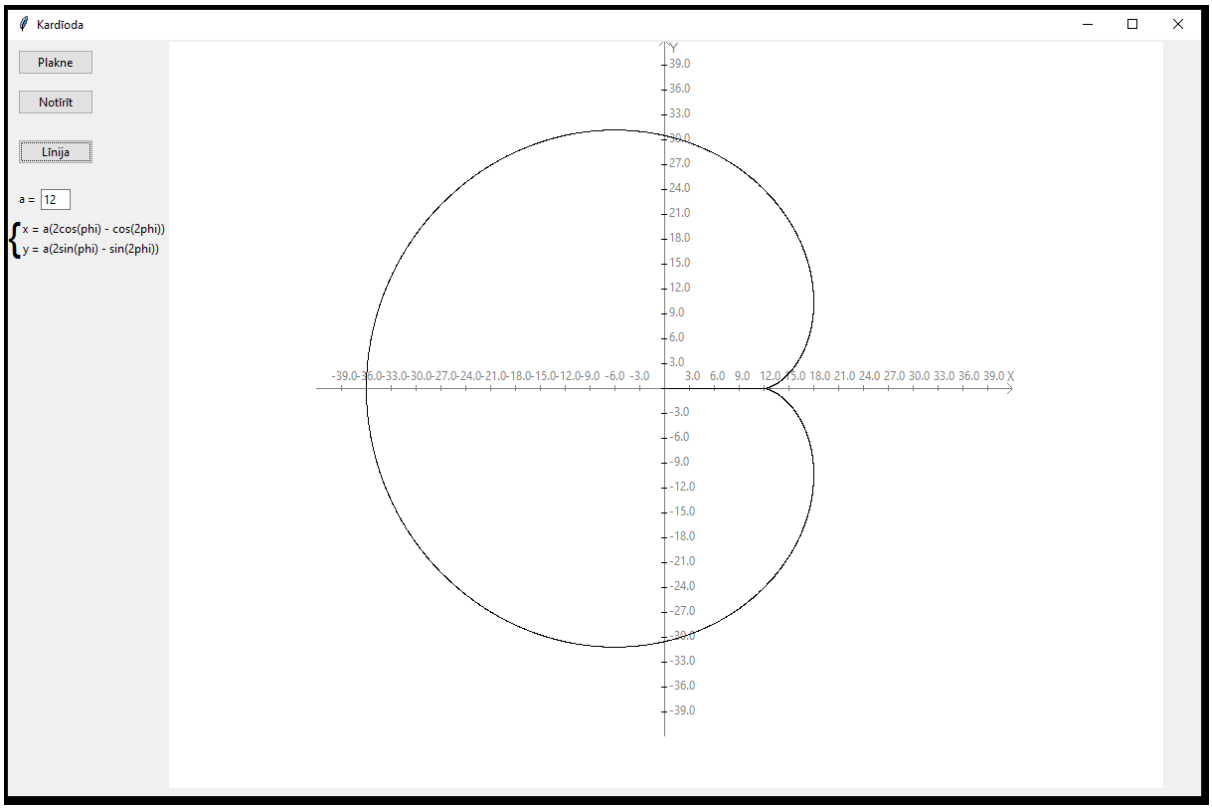
8)


Kardīoda
Plakne
Notīrīt
Līnija
a = -2
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))

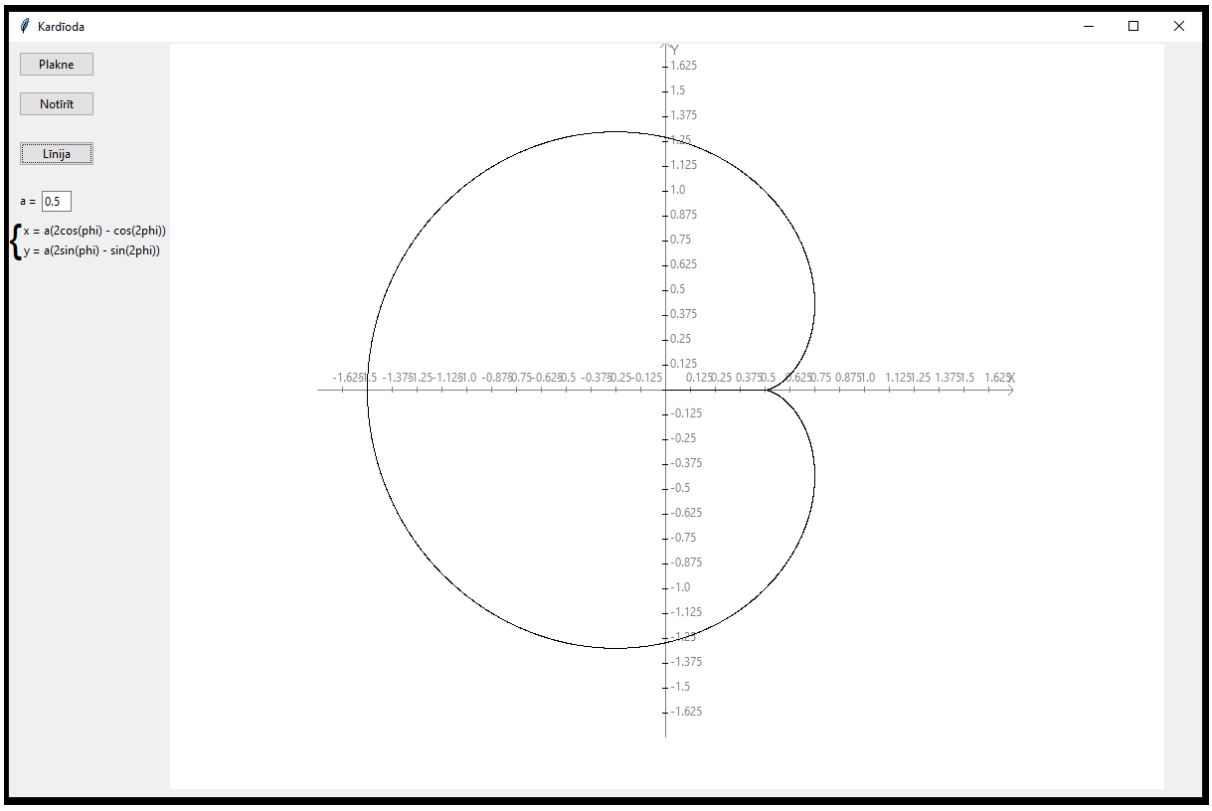
9)


Kardīoda
Plakne
Notīrīt
Līnija
a = divi
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))

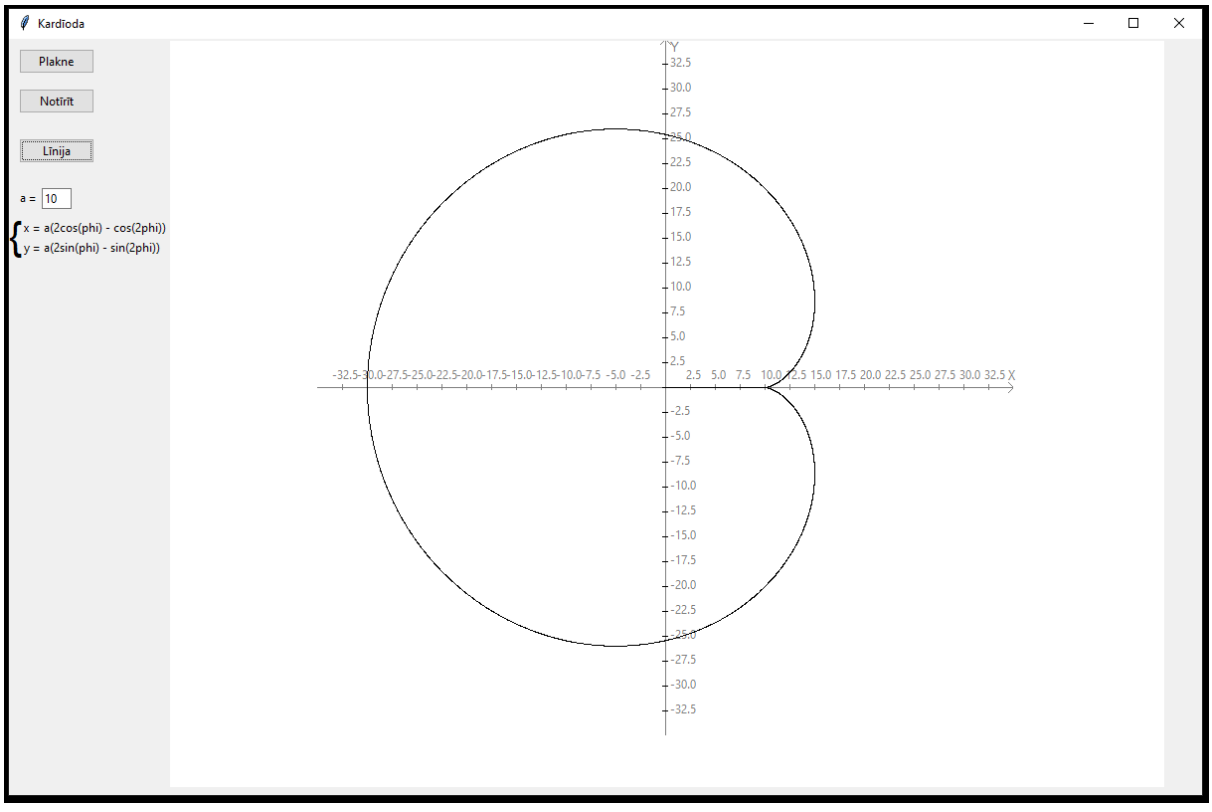
10)


Kardīoda
Plakne
Notīrīt
Līnija
a = 12
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))
39.0
36.0
33.0
30.0
27.0
24.0
21.0
18.0
15.0
12.0
9.0
6.0
3.0
-3.0
-6.0
-9.0
-12.0
-15.0
-18.0
-21.0
-24.0
-27.0
-30.0
-33.0
-36.0
-39.0
-39.0 -36.0 -33.0 -30.0 -27.0 -24.0 -21.0 -18.0 -15.0 -12.0 -9.0 -6.0 -3.0
3.0 6.0 9.0 12.0 15.0 18.0 21.0 24.0 27.0 30.0 33.0 36.0 39.0
X

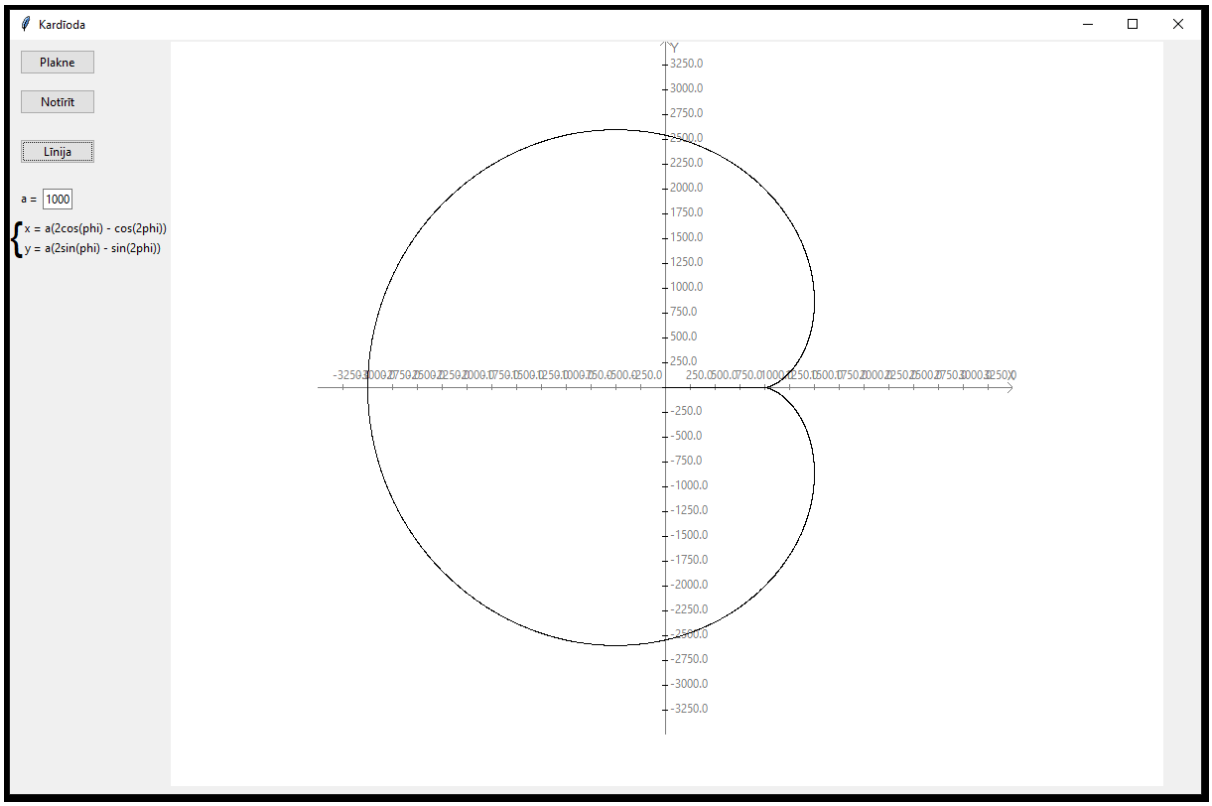
11)


Kardioda
Plakne
Notīrīt
Līnija
a = 0.5
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))

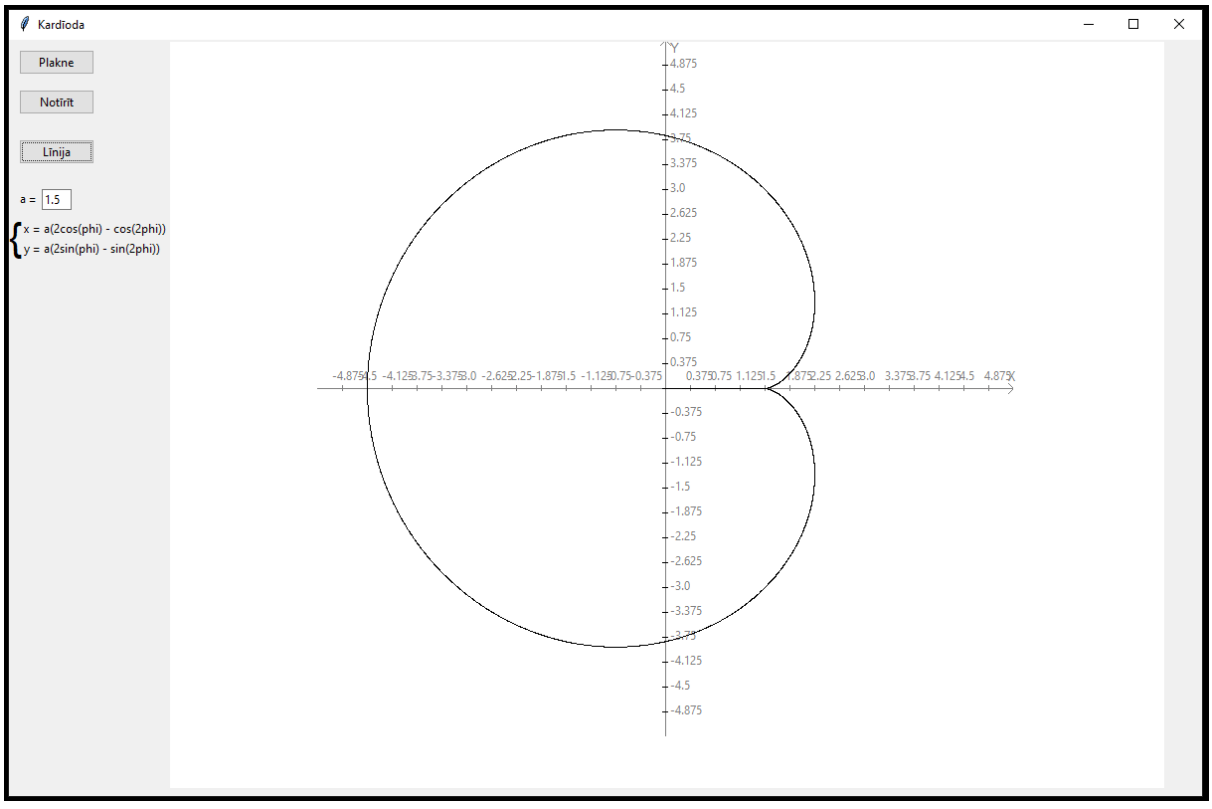
12)


Kardioda
Plakne
Notīrīt
Līnija
a = 10
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))

13)


Kardīoda
Plakne
Notīrīt
Līnija
a = 1000
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))

14)


Kardīoda
Plakne
Notīrīt
Līnija
a = 1.5
x = a(2cos(phi) - cos(2phi))
y = a(2sin(phi) - sin(2phi))