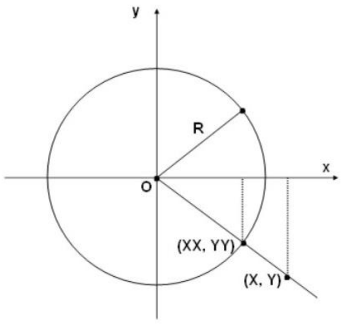
1. uzdevums

Sastādīt programmu, kas nosaka punkta (XX, YY) koordinātas, ja zināms punkta (X, Y) koordinātas un rādiuss R, ko lietotājs ievada no tastatūras. Uzdevums risināms programmu strukturējot izmantojot funkcijas.



Kods:

```
# Programmas nosaukums: 1. uzd MPR15

# 1. uzdevums MPR15

# Uzdevuma formulējums: Sastādīt programmu, kas nosaka punkta (XX, YY) koordinātas, ja zināms punkta (X, Y) koordinātas un rādiuss R, ko lietotājs ievada no tastatūras. Uzdevums risināms programmu strukturējot izmantojot funkcijas.

# Versija 1.0


import math


def is_real_number(x):
    while True:
        try:
            x = float(x)
        except:
            x = input("Mēģini vēlreiz ===> ")
        else:
            return float(x)


def is_real_positive_number(x):
    while True:
        try:
```

```
            x = float(x)
        except:
            x = input("Mēģini vēlreiz ===> ")
        else:
            if x <= 0:
                x = input("Mēģini vēlreiz ===> ")
            else:
                return float(x)


def lidzibas_koeficients(x,y,r):
    hipotenuzes_garums = math.sqrt(x*x + y*y)
    k = r/hipotenuzes_garums # k - līdzības koeficients, r - rādiuss
    return k


def koordinatas_aprekinasana(a,k): # koordinātas aprēķināšana pēc līdzības koeficienta un vienas malas
    return a*k


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


print("Programma nosaka punkta (XX, YY) koordinātas, ja zināms punkta (X, Y) koordinātas un rādiuss R, ko lietotājs ievada no tastatūras.\n")


x = input("Ievadiet X koordinātu ===> ")
x = is_real_number(x)


y = input("Ievadiet Y koordinātu ===> ")
y = is_real_number(y)


if x == 0 and y == 0:
    print("Nav iespējami noteikt")
    quit()
```

```
r = input("Ievadiet rādiusu R ===> ")
r = is_real_positive_number(r)

k = lidzibas_koeficients(x,y,r)

x_koordinatas = koordinatas_aprekinasana(x,k)
y_koordinatas = koordinatas_aprekinasana(y,k)

print("\n(XX, YY) = (" + str(x_koordinatas) + ", " + str(y_koordinatas) + ")")
```

Testa piemēri:

1)

```
Programma nosaka punkta (XX, YY) koordinātas, ja zināms punkta (X, Y) koordinātas un rādiuss R, ko lietotājs ievada no tastatūras.

Ievadiet X koordinātu ===> dasd
Mēģini vēlreiz ===> 12da
Mēģini vēlreiz ===> 6
Ievadiet Y koordinātu ===> dasda
Mēģini vēlreiz ===> asda
Mēģini vēlreiz ===> asdag
Mēģini vēlreiz ===> asfdaqw
Mēģini vēlreiz ===> 8
Ievadiet rādiusu R ===> 5

(XX, YY) = (3.0, 4.0)
```

2)

```
Programma nosaka punkta (XX, YY) koordinātas, ja zināms punkta (X, Y) koordinātas un rādiuss R, ko lietotājs ievada no tastatūras.

Ievadiet X koordinātu ===> 3
Ievadiet Y koordinātu ===> 4
Ievadiet rādiusu R ===> 10

(XX, YY) = (6.0, 8.0)
```

3)

```
Programma nosaka punkta (XX, YY) koordinātas, ja zināms punkta (X, Y) koordinātas un rādiuss R, ko lietotājs ievada no tastatūras.

Ievadiet X koordinātu ===> 0
Ievadiet Y koordinātu ===> 5
Ievadiet rādiusu R ===> 155

(XX, YY) = (0.0, 155.0)
```

4)

```
Programma nosaka punkta (XX, YY) koordinātas, ja zināms punkta (X, Y) koordinātas un rādiuss R, ko lietotājs ievada no tastatūras.

Ievadiet X koordinātu ===> 0
Ievadiet Y koordinātu ===> 0
Nav iespējami noteikt
```

2. uzdevums

Sastādīt programmu, kas pēc ievadīta datuma nosaka, cik dienu ir palicis līdz Ziemassvētku vakaram (24. decembrim). Datuma ievades pieļaujamais formāts ir DD.MM.GGGG. un tas tiek ievadīts kā simbolu virkne. Datuma apstrādes iebūvētās funkcijas izmantot nedrīkst.

Kods:

```
# Programmas nosaukums: 2. uzd. MPR15

# 2. uzdevums MPR15

# Uzdevuma formulējums: Sastādīt programmu, kas pēc ievadīta datuma nosaka, cik dienu ir palicis līdz Ziemassvētku vakaram (24. decembrim). Datuma ievades pieļaujamais formāts ir DD.MM.GGGG. un tas tiek ievadīts kā simbolu virkne. Datuma apstrādes iebūvētās funkcijas izmantot nedrīkst.

# Versija 1.0


def date_check(DD_MM_GGGG_):


    DD_ = DD_MM_GGGG_[0:3]


    MM_ = DD_MM_GGGG_[3:6]


    GGGG_ = DD_MM_GGGG_[6:11]


    DD = DD_MM_GGGG_[0:2]


    MM = DD_MM_GGGG_[3:5]


    GGGG = DD_MM_GGGG_[6:10]


    # ---------- simbolu virknes garums ir precīzi 11 simboli. DD.MM.GGGG.
    if len(DD_MM_GGGG_) != 11:
        return False


    if DD_[2:3] != "." or MM_[2:3] != "." or GGGG_[4:5] != ".":
        return False
```

```
    if DD_MM_GGGG_.count('.') !=3:
        return False

    try:

        DD = int(DD)
        b = 1/DD # 00.MM.GGGG.

        MM = int(MM)
        c = 1/MM # DD.00.GGGG.

        GGGG = int(GGGG)
        d = 1/GGGG # DD.MM.0000.

    except:
        return False
        #print("Tāds datums neeksistē")
        #quit()

    else:
        pass

    DD = int(DD) # Parveidojam int, lai uzzinātu vai tas ir lielāk par 31 vai mazāks vai vienāds ar 0, tad tāds datums neeksistē
    if  DD > 31 or DD <= 0 :
        return False
        #print("Tāds datums neeksistē")
        #quit()

    MM = int(MM) # Parveidojam int, lai uzzinātu vai tas ir lielāk par 12 vai mazāks vai vienāds ar 0, tad tāds datums neeksistē
```

```
    if  MM > 12 or MM <= 0:
        return False
        #print("Tāds datums neeksistē")
        #quit()

    GGGG = int(GGGG)  # Parveidojam int, lai uzzinātu vai tas ir mazāks vai vienāds ar 0, tad tāds datums neeksistē

    if  GGGG <= 0:
        return False
        #print("Tāds datums neeksistē")
        #quit()

    if  MM == 4 and DD > 30: # Aprilī maksimāli ir tikai 30 dienas.
        return False
        #print("Tāds datums neeksistē")
        #quit()

    if  MM == 6 and DD > 30: # Jūnija maksimāli ir tikai 30 dienas.
        return False
        #print("Tāds datums neeksistē")
        #quit()

    if  MM == 9 and DD > 30: # Septembrī maksimāli ir tikai 30 dienas.
        return False
        #print("Tāds datums neeksistē")
        #quit()

    if  MM == 11 and DD > 30: # Novembrī maksimāli ir tikai 30 dienas.
        return False
        #print("Tāds datums neeksistē")
```

```
        #quit()



    F = 28 #default # Dienu skaits Februāri

    if (GGGG % 400) == 0: # Ja garais gad tad februarī dienu skaits ir 29
        F = 29

    elif (GGGG % 100) == 0: # Ja isais gads tad februarī dienu skaits ir 28
        F = 28

    elif (GGGG % 4) == 0: # Ja garais gad tad februarī dienu skaits ir 29
        F = 29

    else:
        F = 28 # Ja isais gads tad februarī dienu skaits ir 28

    if MM == 2 and DD > F: # Lai noteiktu vai ir pareizi ievadīti dati
        return False

def leap_year(GGGG): # Noteicam vai gads (int) ir garais gads vai nav)
    # F = 28 # default
    GGGG = int(GGGG)
    if (GGGG % 400) == 0:
        return True # F = 29

    elif (GGGG % 100) == 0:
        return False # F = 28

    elif (GGGG % 4) == 0:
```

```
        return True # F = 29

    else:
        return False # F = 28



def day_count(Year, MM, DD): # Skaitam cik ir pagajušas dienas no 1. gada līdz ievadītam datumam
    F = 28
    days = 0
    days_year = 0

    days_year = (Year-1)*365 + (Year-1)//4 - (Year-1)//100 + (Year-1)//400 # pagajušo dienu skaits

    if leap_year(GGGG) == False:
        F = 28
    else:
        F = 29

    if MM == "01" :
        days = days_year + DD # only january
        return days
    if MM == "02" : # MM == 2

        days = days_year + DD + 31 # + all of january
        return days
    if MM == "03" : # MM == 3

        days = days_year + DD + 31 + F  # + all of january and february
        return days
    if MM == "04" :
```

```
    days = days_year + DD + 31 + F + 31
    return days
if MM == "05" :

    days = days_year + DD + 31 + F + 31 + 30
    return days
if MM == "06" :

    days = days_year + DD + 31 + F + 31 + 30 + 31
    return days
if MM == "07" :

    days = days_year + DD + 31 + F + 31 + 30 + 31 + 30
    return days
if MM == "08" :

    days = days_year + DD + 31 + F + 31 + 30 + 31 + 30 + 31
    return days
if MM == "09" :

    days = days_year + DD + 31 + F + 31 + 30 + 31 + 30 + 31 + 31
    return days
if MM == "10" :

    days = days_year + DD + 31 + F + 31 + 30 + 31 + 30 + 31 + 31 + 30
    return days
if MM == "11" :

    days = days_year + DD + 31 + F + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31
    return days
if MM == "12" :
```

```
        days = days_year + DD + 31 + F + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30
        return days
    #return days


# ------------------------------------------------- galvenā programma

DD_MM_GGGG_ = input("Ievadiet DD.MM.GGGG. ==> ")

if date_check(DD_MM_GGGG_) == False: # Ja kaut vienu pārbaudi neizturēja, tad tāds datums neeksistē
    print("Tāds datums neeksistē")
    quit()

DD = DD_MM_GGGG_[0:2]
MM = DD_MM_GGGG_[3:5]
GGGG = DD_MM_GGGG_[6:10]

if int(DD) == 24 and MM == "12": # Ja ievada 24.decembrī uzreiz. (vai) (day_count(int(GGGG), str(MM), int(DD))) == (day_count(int(GGGG), str("12"), int(24)))
    print("Šajā dienā ir Ziemassvētku vakars! (24.decembrīs)")

elif (day_count(int(GGGG), str(MM), int(DD))) < (day_count(int(GGGG), str("12"), int(24))):

    remaining_days_until_christmas = (day_count(int(GGGG), str("12"), int(24))) - (day_count(int(GGGG), str(MM), int(DD))) # Ja pirms Ziemāssvetkiem, tad atņemsim cik
    # print(remaining_days_until_christmas)                                    # ir pagājis no 1.gada 1.janvāri līdz šai ievadītam datumam un līdz šīs gāda Ziemāssvētkiem. Tad iegūsim nepieciešamo dienu skaitu
    print("Līdz Ziemassvētkiem ir palikušas " + str(remaining_days_until_christmas) + " dienas.")

else: # Ja (day_count(int(GGGG), str(MM), int(DD))) > (day_count(int(GGGG), str("12"), int(24))):
```

remaining_days_until_christmas = (day_count(int(GGGG)+1, str("12"), int(24))) - (day_count(int(GGGG), str(MM), int(DD))) # Ja pirms Ziemāssvetkiem, tad atņemsim cik ir pagājis no 1.gada 1.janvāri līdz šai ievadītam datumam un līdz šīs gāda+1 (līdz nākama gāda) Ziemāssvētkiem.

print("Līdz Ziemassvētkiem ir palikušas " + str(remaining_days_until_christmas) + " dienas.") # Tad iegūsim nepieciešamo dienu skaitu

Testa piemēri:

1)

```
Ievadiet DD.MM.GGGG. ==> 01.01.0001.
Līdz Ziemassvētkiem ir palikušas 357 dienas.
```

2)

```
Ievadiet DD.MM.GGGG. ==> 24.12.2022.
Šajā dienā ir Ziemassvētki! (24.decembrīs)
```

3)

```
Ievadiet DD.MM.GGGG. ==> 25.12.2022.
Līdz Ziemassvētkiem ir palikušas 364 dienas.
```

4)

```
Ievadiet DD.MM.GGGG. ==> 29.02.2022.
Tāds datums neeksistē
```

5)

```
Ievadiet DD.MM.GGGG. ==> 29.02.2008.
Līdz Ziemassvētkiem ir palikušas 299 dienas.
```

6)

```
Ievadiet DD.MM.GGGG. ==> ad.as.2004.
Tāds datums neeksistē
```

7)

```
Ievadiet DD.MM.GGGG. ==> ..2..2222.
Tāds datums neeksistē
```

8)

```
Ievadiet DD.MM.GGGG. ==> 1231
Tāds datums neeksistē
```

9)

```
Ievadiet DD.MM.GGGG. ==> 66.66.6666.
Tāds datums neeksistē
```

10)

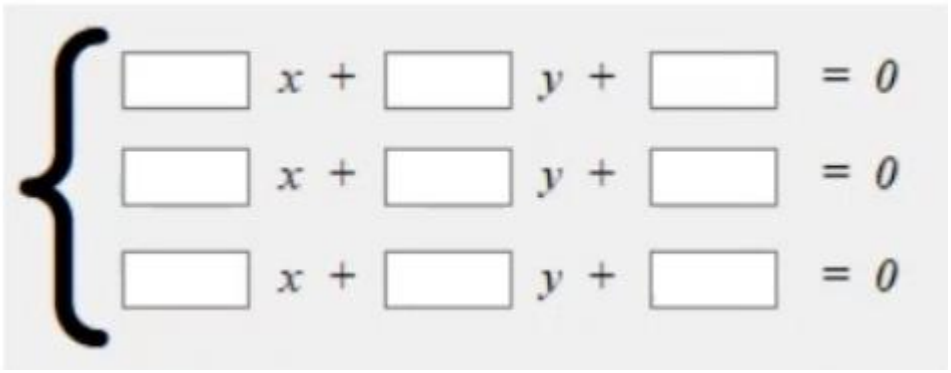
```
Ievadiet DD.MM.GGGG. ==> 1.1.1
Tāds datums neeksistē
```

11)

```
Ievadiet DD.MM.GGGG. ==> 12..12.1222223tsdgwsefg
Tāds datums neeksistē
```

3. uzdevums

Sastādīt programmu, kas noskaidro, vai trīs taisnes, kuras uzdotas ar taišņu vienādojumiem, ir novietotas tā, ka tās veido trijstūri. Ja veido trijstūri, tad aprēķina tā laukumu. Taišņu vienādojumu ievades forma dota attēlā.



Kods:

Programmas nosaukums: 3. uzd MPR15

3. uzdevums MPR15

Uzdevuma formulējums: Sastādīt programmu, kas noskaidro, vai trīs taisnes, kuras uzdotas ar taišņu vienādojumiem, ir novietotas tā, ka tās veido trijstūri. Ja veido trijstūri, tad aprēķina tā laukumu. Taišņu vienādojumu ievades forma dota attēlā.

Versija 1.0

```
# Loga atribūti

import tkinter as tk
from tkinter import * # bez šai rindai nestrādas rinda button['state'] = DISABLED
from tkinter import ttk

def vai_taisnem_ir_kopigs_punkts(A1, B1, A2, B2):
    SD = A1*B2 - A2*B1
    if SD == 0:
        return False
    else:
        return True

def triangle_area_in_coordinates(x1,y1, x2,y2, x3,y3):
    Area = abs((0.5)*(x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2)))
    if Area == 0:
        return False
    else:
        return Area

def two_line_interseption_point_coordinate_x(A1,B1,C1, A2,B2,C2): # (x;y)
    D = A1*B2 - A2*B1
    Dx = -C1*B2 + B1*C2
    x = Dx/D
    return x

def two_line_interseption_point_coordinate_y(A1,B1,C1, A2,B2,C2): # (x;y)
    D = A1*B2 - A2*B1
    Dy = -C2*A1 + A2*C1
    y = Dy/D
```

```
    return y


def triangle_area_in_coordinates(x1,y1, x2,y2, x3,y3):
    Area = abs((0.5)*(x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2)))
    if Area == 0:
        return False
    else:
        return Area


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


'''
A1*x + B1*y + C1 = 0
A2*x + B2*y + C2 = 0
A3*x + B3*y + C3 = 0
'''


def is_realA1(event): # Pārbaudam vai entry e1 ir ierakstīts reāls skaitlīs, nevis parasta simbolu virkne
    global a
    try:
        float(e1.get()) # Pārbaude no e1 ņemsim simbolu virkni un pārbaudīsim vai to var pārveidot float
    except:
        if e1.get() == "":  # ja nekas nav ierakstīts
            e1.config(bg = "white") #  tad iekrasosīm baltā
            l_result.config(text = "") # nodzēsisim ieprēkšējo rezultātu, kuru mēs paradījam lietotājam (aprēķināto laukumu)
            a = False # tad globalais a ir False
        else:
            e1.config(bg = "red")
            l_result.config(text = "")
            a = False
    else:
```

```
        e1.config(bg = "white")
        a = True
    check() # Pārbaudam vai visiem entry ir ierakstīti reāli skaitli


def is_realB1(event):
    global b
    try:
        float(e2.get())
    except:
        if e2.get() == "":
            e2.config(bg = "white")
            l_result.config(text = "")
            b = False
        else:
            e2.config(bg = "red")
            l_result.config(text = "")
            b = False
    else:
        e2.config(bg = "white")
        b = True
    check()


def is_realC1(event):
    global c
    try:
        float(e3.get())
    except:
        if e3.get() == "":
            e3.config(bg = "white")
            l_result.config(text = "")
            c = False
```

```
        else:
            e3.config(bg = "red")
            l_result.config(text = "")
            c = False
    else:
        e3.config(bg = "white")
        c = True
    check()

def is_realA2(event):
    global d
    try:
        float(e4.get())
    except:
        if e4.get() == "":
            e4.config(bg = "white")
            l_result.config(text = "")
            d = False
        else:
            e4.config(bg = "red")
            l_result.config(text = "")
            d = False
    else:
        e4.config(bg = "white")
        d = True
    check()

def is_realB2(event):
    global e
    try:
        float(e5.get())
```

```
    except:
        if e5.get() == "":
            e5.config(bg = "white")
            l_result.config(text = "")
            e = False
        else:
            e5.config(bg = "red")
            l_result.config(text = "")
            e = False
    else:
        e5.config(bg = "white")
        e = True
    check()

def is_realC2(event):
    global f
    try:
        float(e6.get())
    except:
        if e6.get() == "":
            e6.config(bg = "white")
            l_result.config(text = "")
            f = False
        else:
            e6.config(bg = "red")
            l_result.config(text = "")
            f = False
    else:
        e6.config(bg = "white")
        f = True
    check()
```

```
def is_realA3(event):
    global g
    try:
        float(e7.get())
    except:
        if e7.get() == "":
            e7.config(bg = "white")
            l_result.config(text = "")
            g = False
        else:
            e7.config(bg = "red")
            l_result.config(text = "")
            g = False
    else:
        e7.config(bg = "white")
        g = True
    check()

def is_realB3(event):
    global h
    try:
        float(e8.get())
    except:
        if e8.get() == "":
            e8.config(bg = "white")
            l_result.config(text = "")
            h = False
        else:
            e8.config(bg = "red")
            l_result.config(text = "")
```

```
            h = False
        else:
            e8.config(bg = "white")
            h = True
        check()


def is_realC3(event):
    global j
    try:
        float(e9.get())
    except:
        if e9.get() == "":
            e9.config(bg = "white")
            l_result.config(text = "")
            j = False
        else:
            e9.config(bg = "red")
            l_result.config(text = "")
            j = False
    else:
        e9.config(bg = "white")
        j = True
    check()

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


def check(): # Pārbaude
    if a and b and c and d and e and f and g and h and j: # Ja visos lodziņos ir ierakstīti reāli skaitļi
        button['state'] = ACTIVE # tad poga kļūst aktīva
    else:
```

```
        button['state'] = DISABLED # Ja kaut viena lodziņa nav ierakstīts reāls skaitļi, tad poga kļūst aktīva


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


def paraditRezultatu(): # funkcija datu nolasīšanai un apstrādei


    '''
    A1*x + B1*y + C1 = 0
    A2*x + B2*y + C2 = 0
    A3*x + B3*y + C3 = 0
    '''


    A1 = float(e1.get())
    B1 = float(e2.get())
    C1 = float(e3.get())


    A2 = float(e4.get())
    B2 = float(e5.get())
    C2 = float(e6.get())


    A3 = float(e7.get())
    B3 = float(e8.get())
    C3 = float(e9.get())


    # Ja 1. - 2. taisnei ir kopīgs punkts un 2. - 3. taisnei ir kopīgs punkts un 1. - 3. taisnei ir kopīgs punkts, tad atrādīsim visus šos krustpunktu koordinātas
    # lai pēc tam atrisinātu laukumu trīsturim pēc koordinātam
    if vai_taisnem_ir_kopigs_punkts(A1, B1, A2, B2) and vai_taisnem_ir_kopigs_punkts(A2, B2, A3, B3) and vai_taisnem_ir_kopigs_punkts(A1, B1, A3, B3):


        x1 = two_line_interseption_point_coordinate_x(A1,B1,C1, A3,B3,C3) # atradisim x1 koordinatu
```

```
    y1 = two_line_interseption_point_coordinate_y(A1,B1,C1, A3,B3,C3) # atradisim y1 koordinatu

    x2 = two_line_interseption_point_coordinate_x(A2,B2,C2, A3,B3,C3) # atradisim x2 koordinatu
    y2 = two_line_interseption_point_coordinate_y(A2,B2,C2, A3,B3,C3) # atradisim y2 koordinatu

    x3 = two_line_interseption_point_coordinate_x(A1,B1,C1, A2,B2,C2) # atradisim x3 koordinatu
    y3 = two_line_interseption_point_coordinate_y(A1,B1,C1, A2,B2,C2) # atradisim y3 koordinatu

    S = triangle_area_in_coordinates(x1,y1, x2,y2, x3,y3) # trijstura laukums koordinātas

    if S != 0:
      l_result.config(text = "S = " + str(S)) # Izvadam rezultātu
    else:
      l_result.config(text = "Neveidojas trijstūris.") # Ja laukums ir 0, tad tas ir gadījums, kad visas taisnēs krustojās vienā punktā
  else:
    l_result.config(text = "Neveidojas trijstūris.") # Ja kaut viena prasība neizpildās, tad neveidojas trījstūris

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

# Sākuma visi globālie mainīgie ir False. Viņi atbild par to, vai poga ( button = ttk.Button(logs, text="=", width=1, command=paraditRezultatu) )
# ir ieslēgta vai izslēgta

a = False
b = False
c = False
d = False
e = False
f = False
g = False
```

```
h = False
j = False

logs = tk.Tk()
logs.title("Taišņu veidotais laukums") # Programmas nosaukums
logs.geometry("600x300")
l_result = ttk.Label(logs, text="")
l_result.place(x=80, y=200)



# etiķešu (uzrakstu) iezveide

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

l_iekava = ttk.Label(logs, text="{", font = ("Calibri", 140))
l_iekava.place(x=-5, y=-20)

l1 = ttk.Label(logs, text="x + ")
l1.place(x=95, y=50)

l2 = ttk.Label(logs, text="y + ")
l2.place(x=170, y=50)

l0_1 = ttk.Label(logs, text=" =  0 ")
l0_1.place(x=245, y=50)

# ----------------------
# ----------------------
l3 = ttk.Label(logs, text="x + ")
l3.place(x=95, y=100)
```

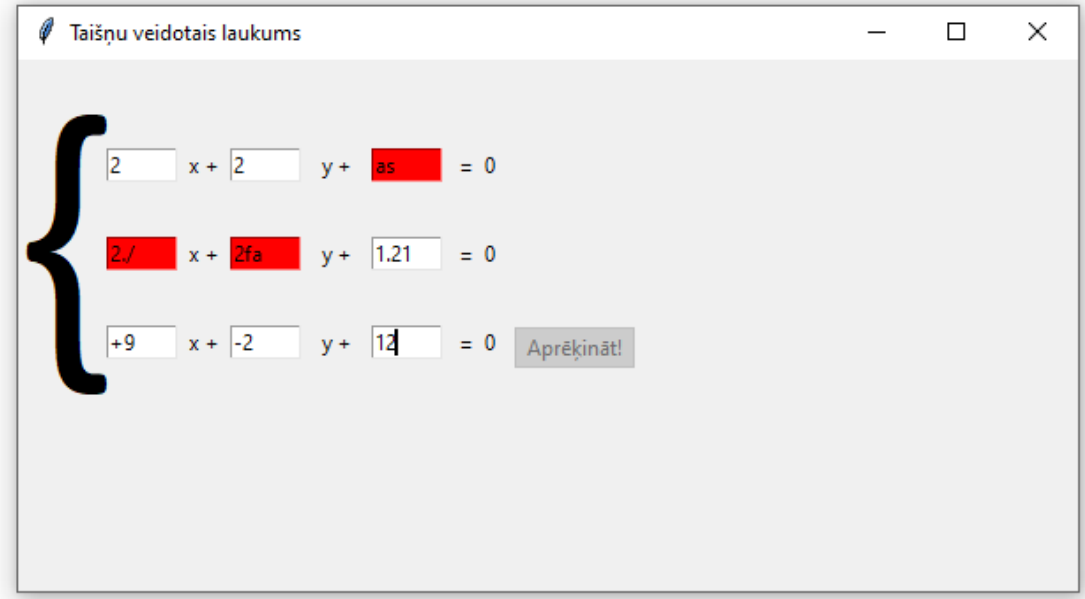
```
l4 = ttk.Label(logs, text="y + ")
l4.place(x=170, y=100)

l0_2 = ttk.Label(logs, text=" =  0 ")
l0_2.place(x=245, y=100)
# ----------------------

l5 = ttk.Label(logs, text="x + ")
l5.place(x=95, y=150)

l6 = ttk.Label(logs, text="y + ")
l6.place(x=170, y=150)

l0_3 = ttk.Label(logs, text=" =  0 ")
l0_3.place(x=245, y=150)
# ----------------------



# ievades tektlodziņu izveide

# ----------------------
e1 = tk.Entry(logs, width = 6)
e1.bind("<KeyRelease>", is_realA1) # Izsaucam funkciju is_realA1
e1.place(x=50, y=50)

e2 = tk.Entry(logs, width = 6)
e2.bind("<KeyRelease>", is_realB1 )
e2.place(x=120, y=50)

e3 = tk.Entry(logs, width = 6)
e3.bind("<KeyRelease>", is_realC1 )
```

```
e3.place(x=200, y=50)
# ----------------------


# ----------------------
e4 = tk.Entry(logs, width = 6)
e4.bind("<KeyRelease>", is_realA2 )
e4.place(x=50, y=100)

e5 = tk.Entry(logs, width = 6)
e5.bind("<KeyRelease>", is_realB2 )
e5.place(x=120, y=100)

e6 = tk.Entry(logs, width = 6)
e6.bind("<KeyRelease>", is_realC2 )
e6.place(x=200, y=100)
# ----------------------


# ----------------------
e7 = tk.Entry(logs, width = 6)
e7.bind("<KeyRelease>", is_realA3 )
e7.place(x=50, y=150)

e8 = tk.Entry(logs, width = 6)
e8.bind("<KeyRelease>", is_realB3 )
e8.place(x=120, y=150)

e9 = tk.Entry(logs, width = 6)
e9.bind("<KeyRelease>", is_realC3 )
e9.place(x=200, y=150)
# ----------------------
```

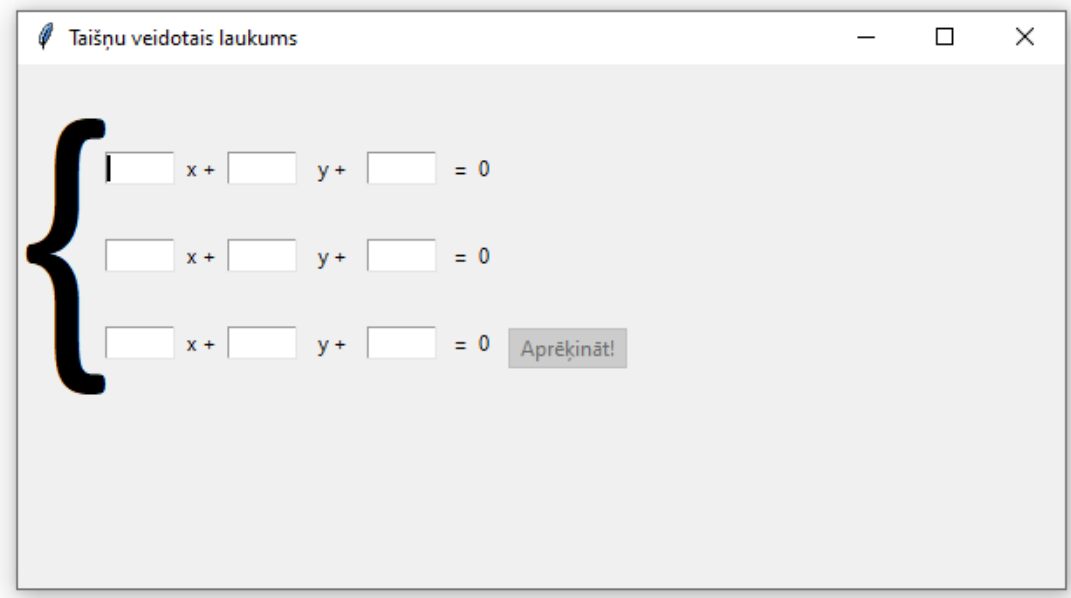
```
button = ttk.Button(logs, text="Aprēķināt!", width=10, command=paraditRezultatu)
button.place(x=280, y=150) # novietojām pogu koordinātas x = 250 y = 150
button['state'] = DISABLED # pēc noklusējuma poga ir izslēgta

# Obligāta rindiņa, lai logs būtu redzāms visu laiku
logs.mainloop()
```

Testa piemēri:

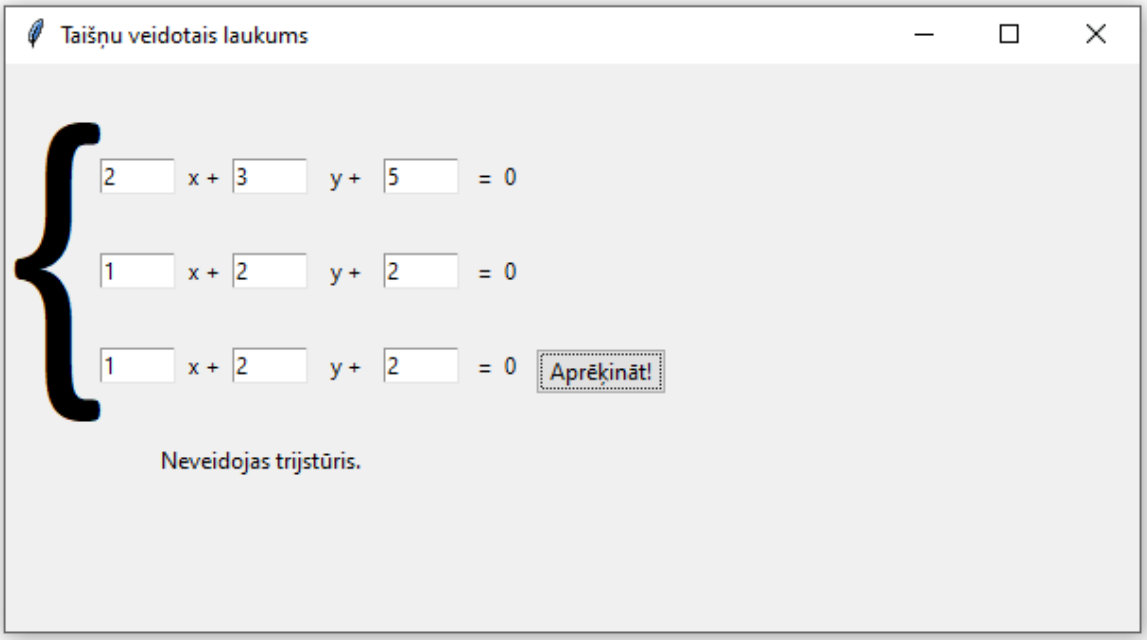
1) Ja nav pareizi ievadīti dati, tad nav iespējams nospiest uz pogu aprēķināt



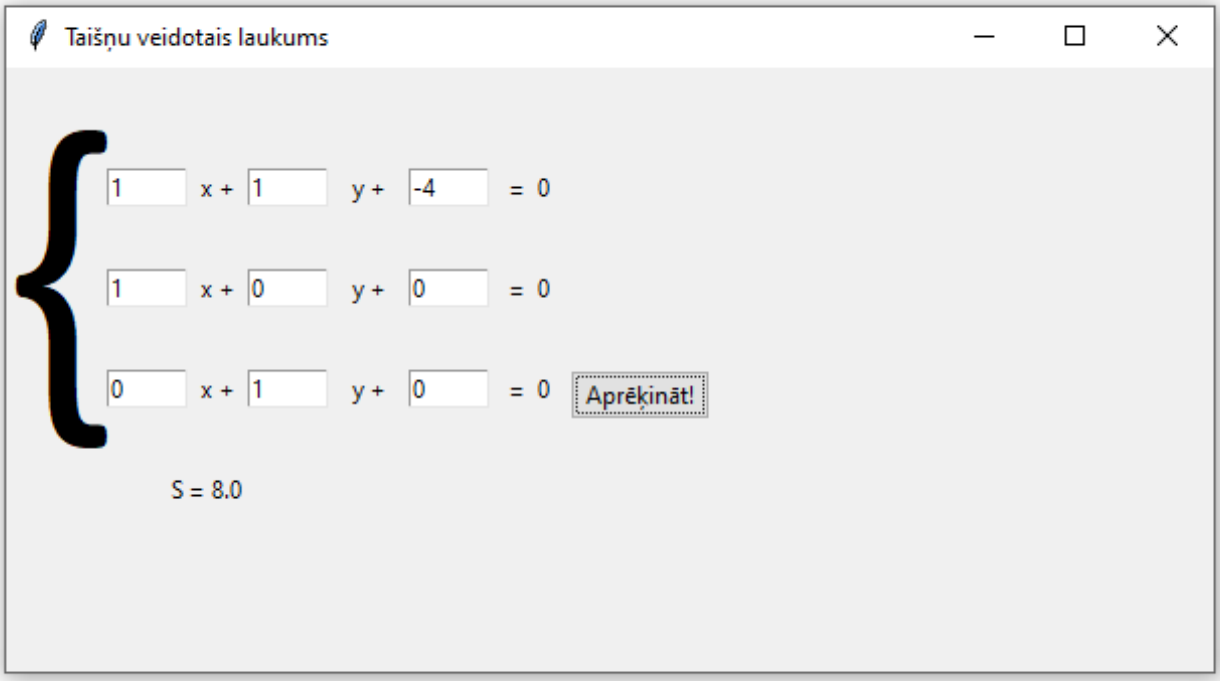
2)



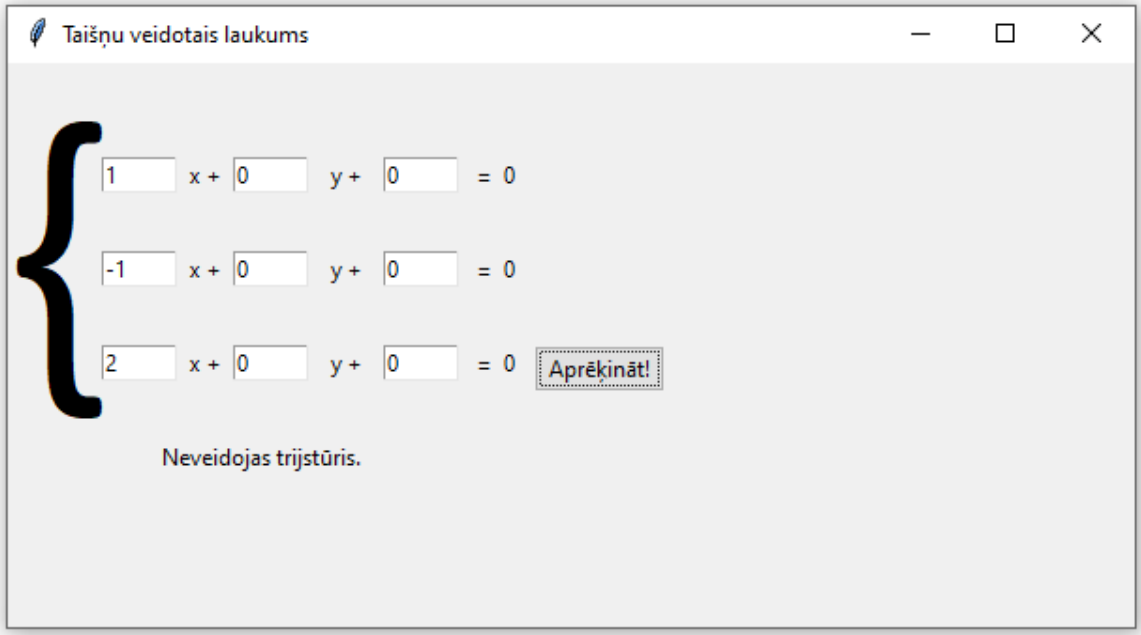
3)


Taišņu veidotais laukums
2 x + 3 y + 5 = 0
1 x + 2 y + 2 = 0
1 x + 2 y + 2 = 0
Aprēķināt!
Neveidojas trijstūris.

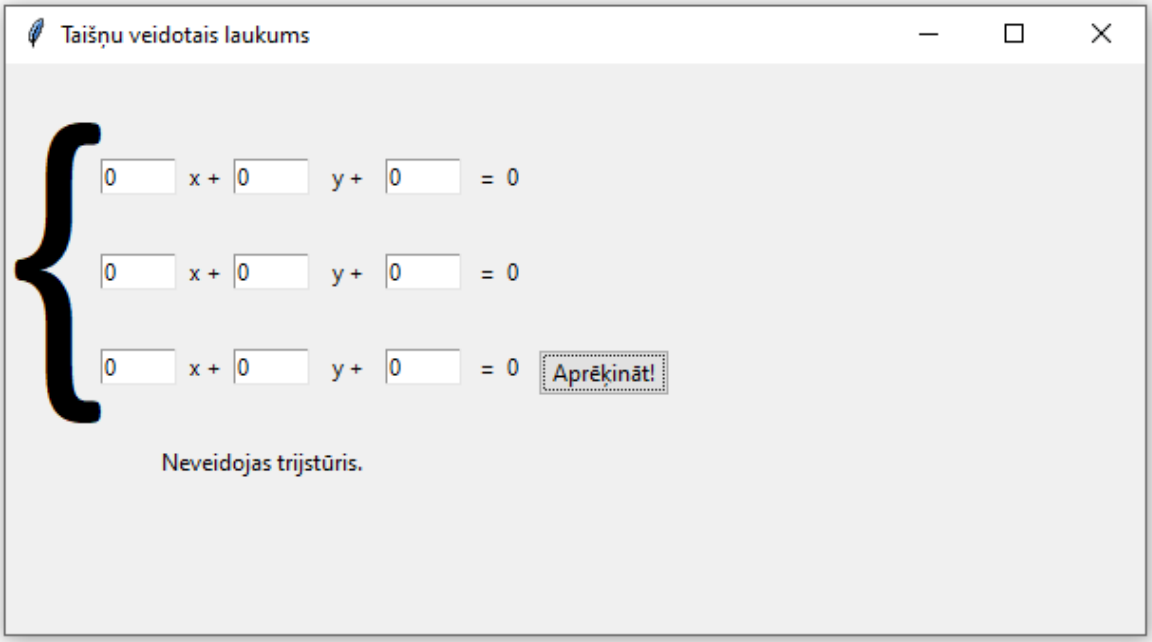
4)


Taišņu veidotais laukums
1 x + 1 y + -4 = 0
1 x + 0 y + 0 = 0
0 x + 1 y + 0 = 0
Aprēķināt!
S = 8.0

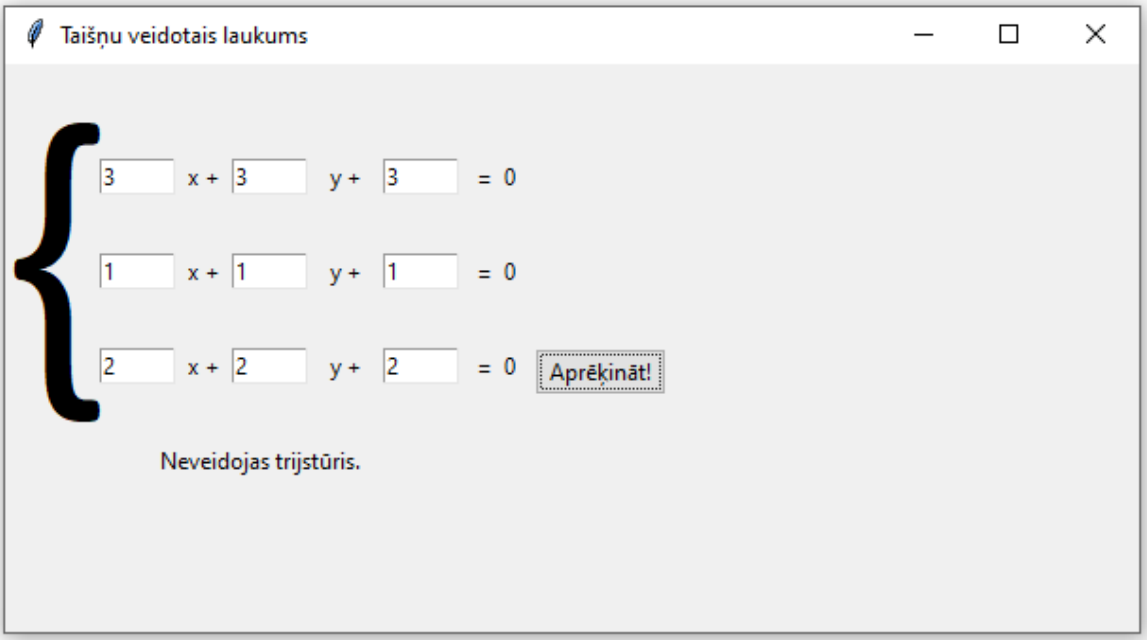
5)


Taišņu veidotais laukums
1 x + 0 y + 0 = 0
-1 x + 0 y + 0 = 0
2 x + 0 y + 0 = 0
Aprēķināt!
Neveidojas trijstūris.

6)


Taišņu veidotais laukums
0 x + 0 y + 0 = 0
0 x + 0 y + 0 = 0
0 x + 0 y + 0 = 0
Aprēķināt!
Neveidojas trijstūris.

7)



3. uzdevums. PU1

Sastādīt programmu, kas noskaidro, vai trīs taisnes, kuras uzdotas ar taišņu vienādojumiem, ir novietotas tā, ka tās veido trijstūri. Ja veido trijstūri, tad aprēķina tā laukumu. Taišņu vienādojumu ievades forma dota attēlā. Uzzīmēt taisnes koordinātu plaknē, ievērojot mērogu. (Izveidota programmā tika uzzīmēti nogriežņi, kuri savieno trijstūra virsotnes, jo tas uzskatamāk).

Kods:

```
# Programmas nosaukums: 3. uzd. MPR15

# 3. uzdevums MPR15

# Uzdevuma formulējums: Sastādīt programmu, kas noskaidro, vai trīs taisnes, kuras uzdotas ar taišņu vienādojumiem, ir novietotas tā, ka tās veido trijstūri. Ja veido trijstūri, tad aprēķina tā laukumu. Taišņu vienādojumu ievades forma dota attēlā.

# Versija 1.0
```

```
# Loga atribūti

import tkinter as tk
from tkinter import *
from tkinter import ttk

def enable_button(): # Aktīve pogu "Parādīt funkciju"
    button['state'] = ACTIVE

def disable_button(): # Dizaktīve pogu "Parādīt funkciju"
    button['state'] = DISABLED

def notirit(): # funkcija kanva zīmejuma dzēšanai

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

def paradit(): # funkcija koordinātu plaknes 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 vai_taisnem_ir_kopigs_punkts(A1, B1, A2, B2): # Noskaidro vai divām taisnēm ir kopīgs punkts vai nav
    SD = A1*B2 - A2*B1
    if SD == 0: # Sistēmas determinants
        return False # Ja sistēmas determinants ir vienāds ar nulli, tad taisnes nekrustojas, tad tam nav kopīga punkta (return False)
    else:
        return True

def triangle_area_in_coordinates(x1,y1, x2,y2, x3,y3): # Noskaidro trijstūra laukumu pēc virsotņu koordinātam
    Area = abs((0.5)*(x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2)))
```

```
    if Area == 0:
        return False
    else:
        return Area

def two_line_interseption_point_coordinate_x(A1,B1,C1, A2,B2,C2): # (x;y)
    D = A1*B2 - A2*B1
    Dx = -C1*B2 + B1*C2
    x = Dx/D
    return x # Atrod divas taisnes krustpunktu (x koordināta)

def two_line_interseption_point_coordinate_y(A1,B1,C1, A2,B2,C2): # (x;y)
    D = A1*B2 - A2*B1
    Dy = -C2*A1 + A2*C1
    y = Dy/D
    return y # Atrod divas taisnes krustpunktu (y koordināta)



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

'''
A1*x + B1*y + C1 = 0
A2*x + B2*y + C2 = 0
A3*x + B3*y + C3 = 0
'''

def is_realA1(event): # Pārbaudam vai entry e1 ir ierakstīts reāls skaitlīs, nevis parasta simbolu virkne
    global a
    try:
        float(e1.get()) # Pārbaude no e1 ņemsim simbolu virkni un pārbaudīsim vai to var pārveidot float
```

```
    except:
        if e1.get() == "":  # ja nekas nav ierakstīts
            e1.config(bg = "white") #  tad iekrasosīm baltā
            l_result.config(text = "") # nodzēsisim ieprēkšējo rezultātu, kuru mēs paradījam lietotājam (aprēķināto laukumu)
            a = False # tad globalais a ir False
        else:
            e1.config(bg = "red")
            l_result.config(text = "")
            a = False
    else:
        e1.config(bg = "white")
        a = True
    check() # Pārbaudam vai visiem entry ir ierakstīti reāli skaitli


def is_realB1(event):
    global b
    try:
        float(e2.get())
    except:
        if e2.get() == "":
            e2.config(bg = "white")
            l_result.config(text = "")
            b = False
        else:
            e2.config(bg = "red")
            l_result.config(text = "")
            b = False
    else:
        e2.config(bg = "white")
        b = True
    check()
```

```
def is_realC1(event):
    global c
    try:
        float(e3.get())
    except:
        if e3.get() == "":
            e3.config(bg = "white")
            l_result.config(text = "")
            c = False
        else:
            e3.config(bg = "red")
            l_result.config(text = "")
            c = False
    else:
        e3.config(bg = "white")
        c = True
    check()

def is_realA2(event):
    global d
    try:
        float(e4.get())
    except:
        if e4.get() == "":
            e4.config(bg = "white")
            l_result.config(text = "")
            d = False
        else:
            e4.config(bg = "red")
            l_result.config(text = "")
```

```
            d = False
        else:
            e4.config(bg = "white")
            d = True
        check()

def is_realB2(event):
    global e
    try:
        float(e5.get())
    except:
        if e5.get() == "":
            e5.config(bg = "white")
            l_result.config(text = "")
            e = False
        else:
            e5.config(bg = "red")
            l_result.config(text = "")
            e = False
    else:
        e5.config(bg = "white")
        e = True
    check()

def is_realC2(event):
    global f
    try:
        float(e6.get())
    except:
        if e6.get() == "":
            e6.config(bg = "white")
```

```
            l_result.config(text = "")
            f = False
        else:
            e6.config(bg = "red")
            l_result.config(text = "")
            f = False
    else:
        e6.config(bg = "white")
        f = True
    check()

def is_realA3(event):
    global g
    try:
        float(e7.get())
    except:
        if e7.get() == "":
            e7.config(bg = "white")
            l_result.config(text = "")
            g = False
        else:
            e7.config(bg = "red")
            l_result.config(text = "")
            g = False
    else:
        e7.config(bg = "white")
        g = True
    check()

def is_realB3(event):
    global h
```

```
    try:
        float(e8.get())
    except:
        if e8.get() == "":
            e8.config(bg = "white")
            l_result.config(text = "")
            h = False
        else:
            e8.config(bg = "red")
            l_result.config(text = "")
            h = False
    else:
        e8.config(bg = "white")
        h = True
    check()


def is_realC3(event):
    global j
    try:
        float(e9.get())
    except:
        if e9.get() == "":
            e9.config(bg = "white")
            l_result.config(text = "")
            j = False
        else:
            e9.config(bg = "red")
            l_result.config(text = "")
            j = False
    else:
```

```
        e9.config(bg = "white")
        j = True
    check()

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

def check():
    if a and b and c and d and e and f and g and h and j:
        button['state'] = ACTIVE
    else:
        button['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 paraditRezultatu(): # funkcija datu nolasīšanai un apstrādei
    paradit()

    A1 = float(e1.get())
    B1 = float(e2.get())
    C1 = float(e3.get())

    A2 = float(e4.get())
    B2 = float(e5.get())
    C2 = float(e6.get())

    A3 = float(e7.get())
    B3 = float(e8.get())
    C3 = float(e9.get())

    if vai_taisnem_ir_kopigs_punkts(A1, B1, A2, B2) and vai_taisnem_ir_kopigs_punkts(A2, B2, A3, B3)
and vai_taisnem_ir_kopigs_punkts(A1, B1, A3, B3):
```

```
    #global x1,y1,x2,y2,x3,y3
    x1 = two_line_interseption_point_coordinate_x(A1,B1,C1, A3,B3,C3) # atradisim x1 koordinatu
    y1 = two_line_interseption_point_coordinate_y(A1,B1,C1, A3,B3,C3) # atradisim y1 koordinatu

    x2 = two_line_interseption_point_coordinate_x(A2,B2,C2, A3,B3,C3) # atradisim x2 koordinatu
    y2 = two_line_interseption_point_coordinate_y(A2,B2,C2, A3,B3,C3) # atradisim y2 koordinatu

    x3 = two_line_interseption_point_coordinate_x(A1,B1,C1, A2,B2,C2) # atradisim x3 koordinatu
    y3 = two_line_interseption_point_coordinate_y(A1,B1,C1, A2,B2,C2) # atradisim y3 koordinatu

    #print(x1) koordinātas pārbaudīšānai
    #print(y1)
    #print(x2)
    #print(y2)
    #print(x3)
    #print(y3)

    # Savienojam iegutus krustpunktus (1.-3., 2.-3., 1.-3.) (numuri. Pirmā taisne ar trešo, otrā taisne ar trešo, un prmā taisne ar trēšo)

    kanva.create_line(x1*25 + 500, 350 - y1*25,    x2*25 + 500, 350 - y2*25) # 500 un 350 tas ir koordinātu centrs
    kanva.create_line(x2*25 + 500, 350 - y2*25,    x3*25 + 500, 350 - y3*25) # mērogs ir tāds, ka lai iegūtu pareizo zimējumu ir jāreizina ar 25
    kanva.create_line(x1*25 + 500, 350 - y1*25,    x3*25 + 500, 350 - y3*25)

    S = triangle_area_in_coordinates(x1,y1, x2,y2, x3,y3) # trijstura laukums koordinātas

    if S != 0:
       l_result.config(text = "S = " + str(S)) # parādam lietotājam laukumu
    else:
       notirit()
```

```
        l_result.config(text = "Neveidojas trijstūris.") # Ja laukums ir 0, tad tas ir gadījums, kad visas taisnēs krustojās vienā punktā. Tad trijstūris neveidojas.
    else:
        notirit()
        l_result.config(text = "Neveidojas trijstūris.") # Ja kaut viena prasība neizpildās, tad neveidojas trījstūris


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


def zimesana(x1,y1,x2,y2,x3,y3):
    # (500 , 350)  centrs
    kanva.create_line(x1 + 500 , y1 + 350, x2 + 500 , y2 + 350)
    kanva.create_line(x2 + 500 , y2 + 350, x3 + 500 , y3 + 350)
    kanva.create_line(x1 + 500, y1 + 350, x3 + 500, y3 + 350)


# Sākuma visi globālie mainīgie ir False. Viņi atbild par to, vai poga ( button = ttk.Button(logs, text="=", width=1, command=paraditRezultatu) )
# ir ieslēgta vai izslēgta

a = False
b = False
c = False
d = False
e = False
f = False
g = False
h = False
j = False

x1 = 0
y1 = 0
x2 = 0
```

```
y2 = 0
x3 = 0
y3 = 0

logs = tk.Tk()
logs.title("Taišņu veidotais laukums")
logs.geometry("1200x800")

kanva = tk.Canvas(logs, bg="white", height=10000, width=10860)
kanva.place(x=300, y=40)

l_result = ttk.Label(logs, text="")
l_result.place(x=80, y=250)


poga2 = ttk.Button(logs, text="Notīrīt", command=notirit)
poga2.place(x=108, y=200)


# etiķešu (uzrakstu) iezveide

# ----------------------
l_iekava = ttk.Label(logs, text="{", font = ("Calibri", 140))
l_iekava.place(x=-5, y=-20)

l1 = ttk.Label(logs, text="x + ")
l1.place(x=95, y=50)

l2 = ttk.Label(logs, text="y + ")
l2.place(x=170, y=50)
```

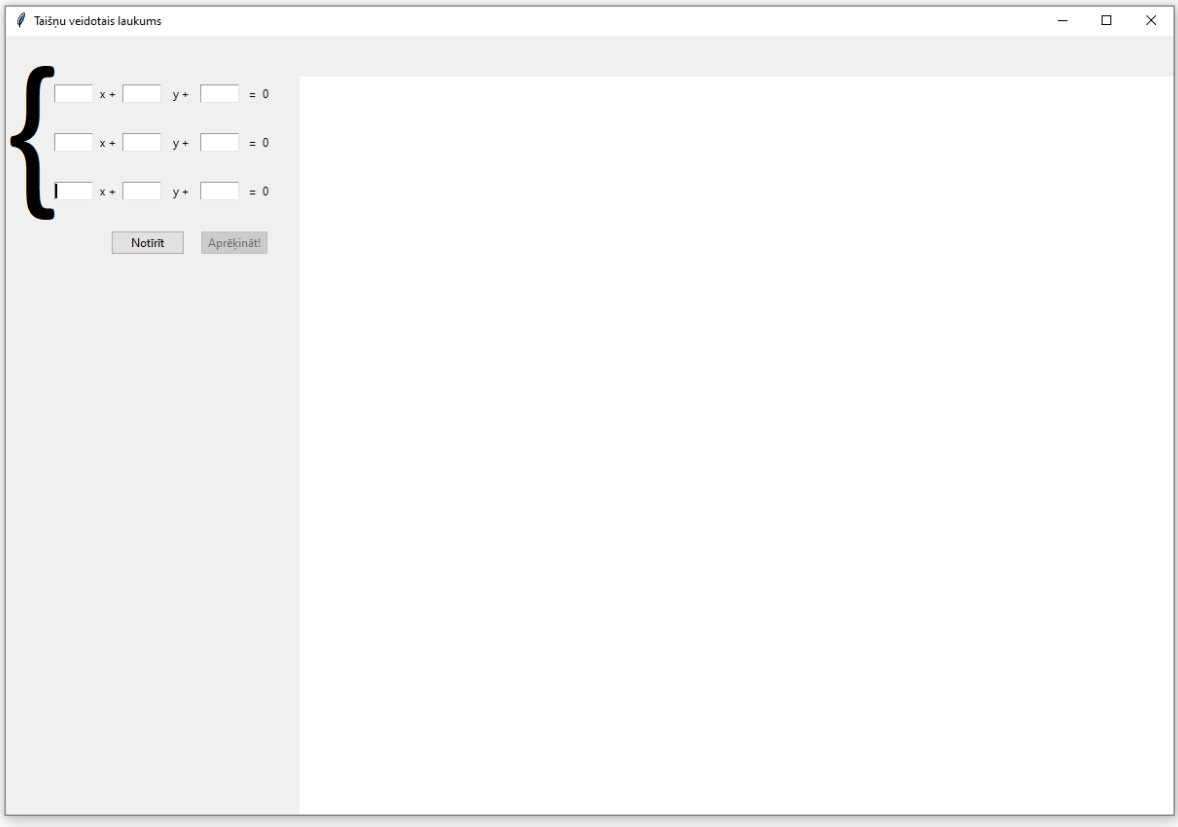
```
l0_1 = ttk.Label(logs, text=" =  0 ")
l0_1.place(x=245, y=50)

# ----------------------
# ----------------------
l3 = ttk.Label(logs, text="x + ")
l3.place(x=95, y=100)

l4 = ttk.Label(logs, text="y + ")
l4.place(x=170, y=100)

l0_2 = ttk.Label(logs, text=" =  0 ")
l0_2.place(x=245, y=100)
# ----------------------

l5 = ttk.Label(logs, text="x + ")
l5.place(x=95, y=150)

l6 = ttk.Label(logs, text="y + ")
l6.place(x=170, y=150)

l0_3 = ttk.Label(logs, text=" =  0 ")
l0_3.place(x=245, y=150)
# ----------------------


# ievades tektlodziņu izveide

# ----------------------
e1 = tk.Entry(logs, width = 6)
e1.bind("<KeyRelease>", is_realA1) # Izsaucam funkciju is_realA1
```

```
e1.place(x=50, y=50)

e2 = tk.Entry(logs, width = 6)
e2.bind("<KeyRelease>", is_realB1 )
e2.place(x=120, y=50)

e3 = tk.Entry(logs, width = 6)
e3.bind("<KeyRelease>", is_realC1 )
e3.place(x=200, y=50)
# ----------------------

# ----------------------
e4 = tk.Entry(logs, width = 6)
e4.bind("<KeyRelease>", is_realA2 )
e4.place(x=50, y=100)

e5 = tk.Entry(logs, width = 6)
e5.bind("<KeyRelease>", is_realB2 )
e5.place(x=120, y=100)

e6 = tk.Entry(logs, width = 6)
e6.bind("<KeyRelease>", is_realC2 )
e6.place(x=200, y=100)
# ----------------------

# ----------------------
e7 = tk.Entry(logs, width = 6)
e7.bind("<KeyRelease>", is_realA3 )
e7.place(x=50, y=150)

e8 = tk.Entry(logs, width = 6)
```

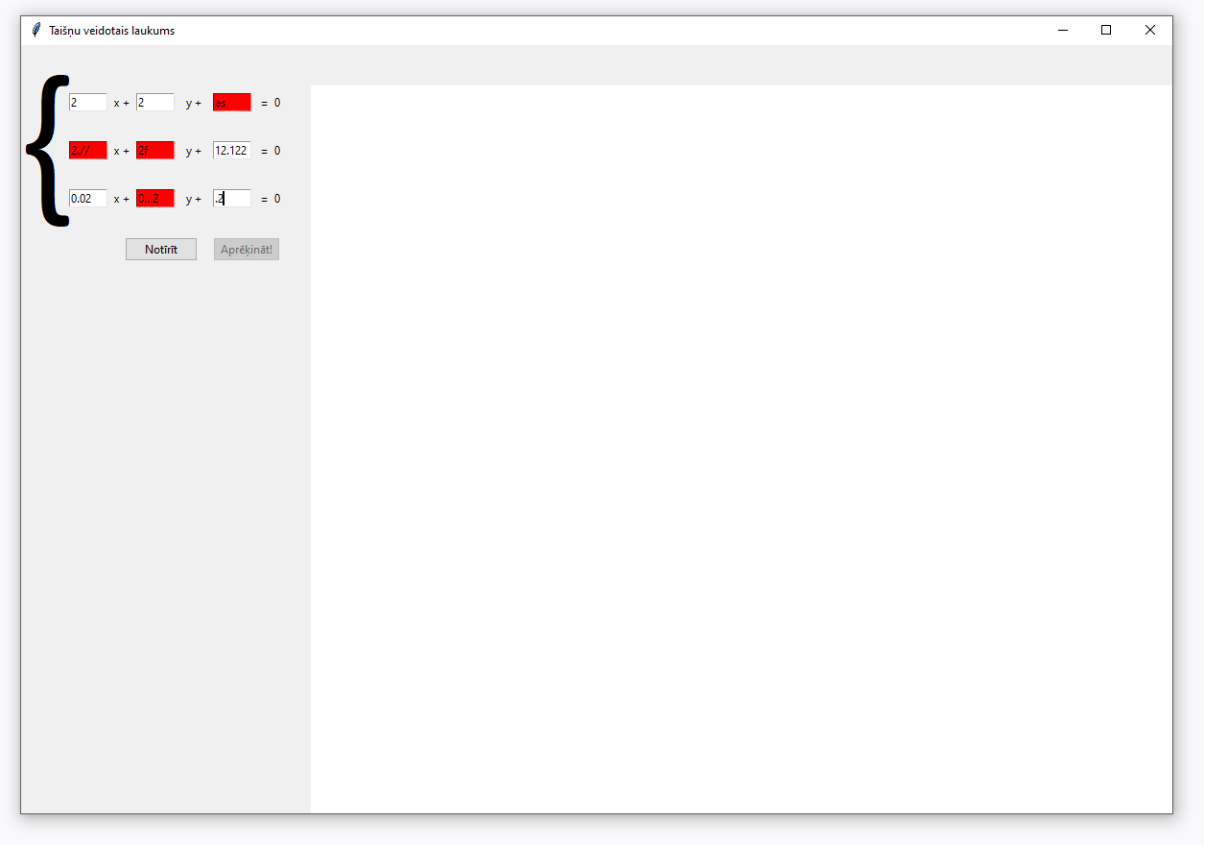
```
e8.bind("<KeyRelease>", is_realB3 )
e8.place(x=120, y=150)

e9 = tk.Entry(logs, width = 6)
e9.bind("<KeyRelease>", is_realC3 )
e9.place(x=200, y=150)
# ----------------------

button = ttk.Button(logs, text="Aprēķināt!", width=10, command=paraditRezultatu)
button.place(x=200, y=200) # novietojām pogu koordinātas x = 250 y = 150
button['state'] = DISABLED # pēc noklusējuma poga ir izslēgta

# Obligāta rindiņa, lai logs būtu redzāms visu laiku
logs.mainloop()
```

Testa piemēri:

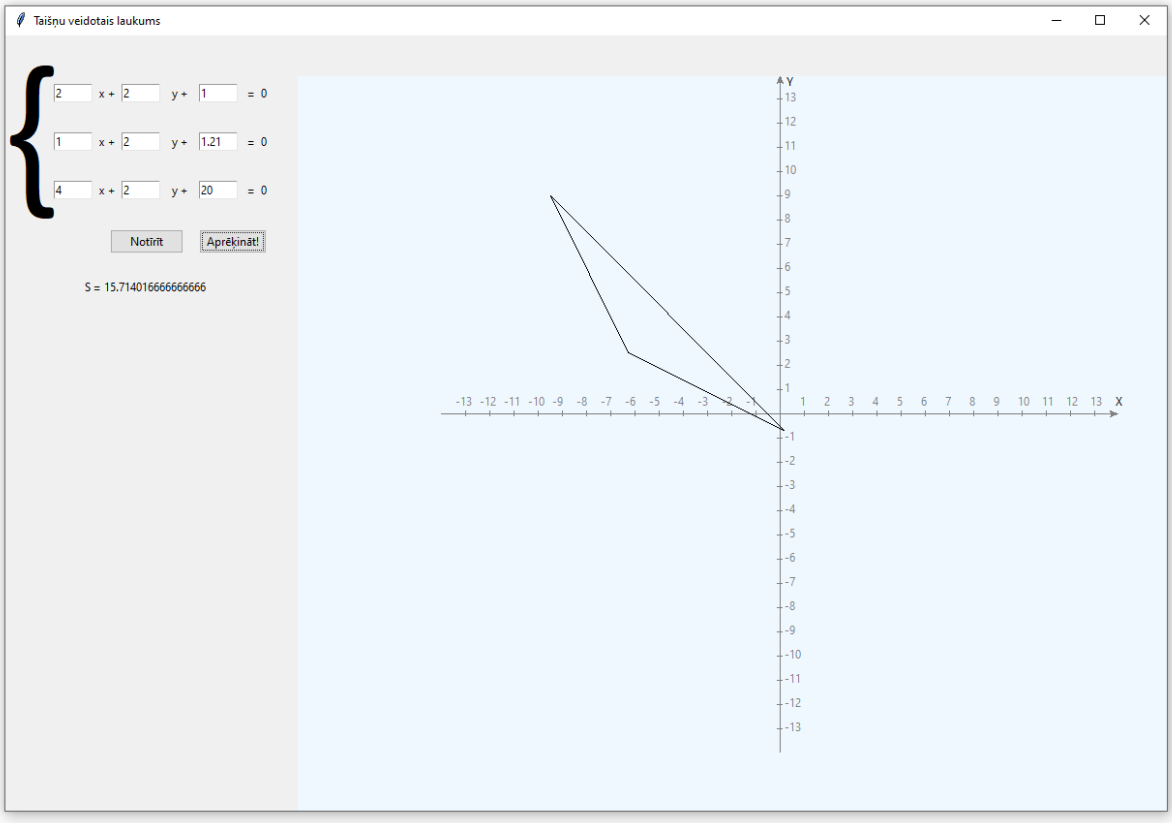
1)



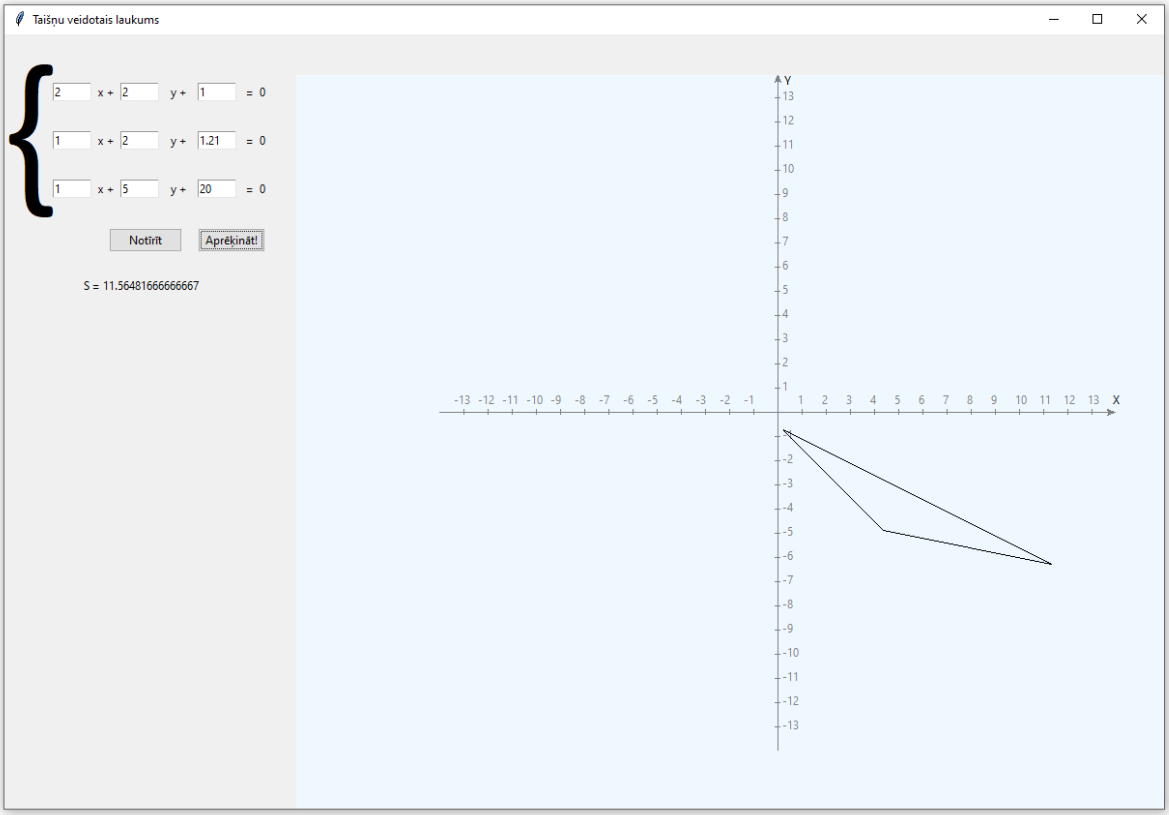
2)


Taišņu veidotais laukums
2 x + 2 y + as = 0
2.// x + 2f y + 12.122 = 0
0.02 x + 0...2 y + .4 = 0
Notīrīt
Aprēķināt!

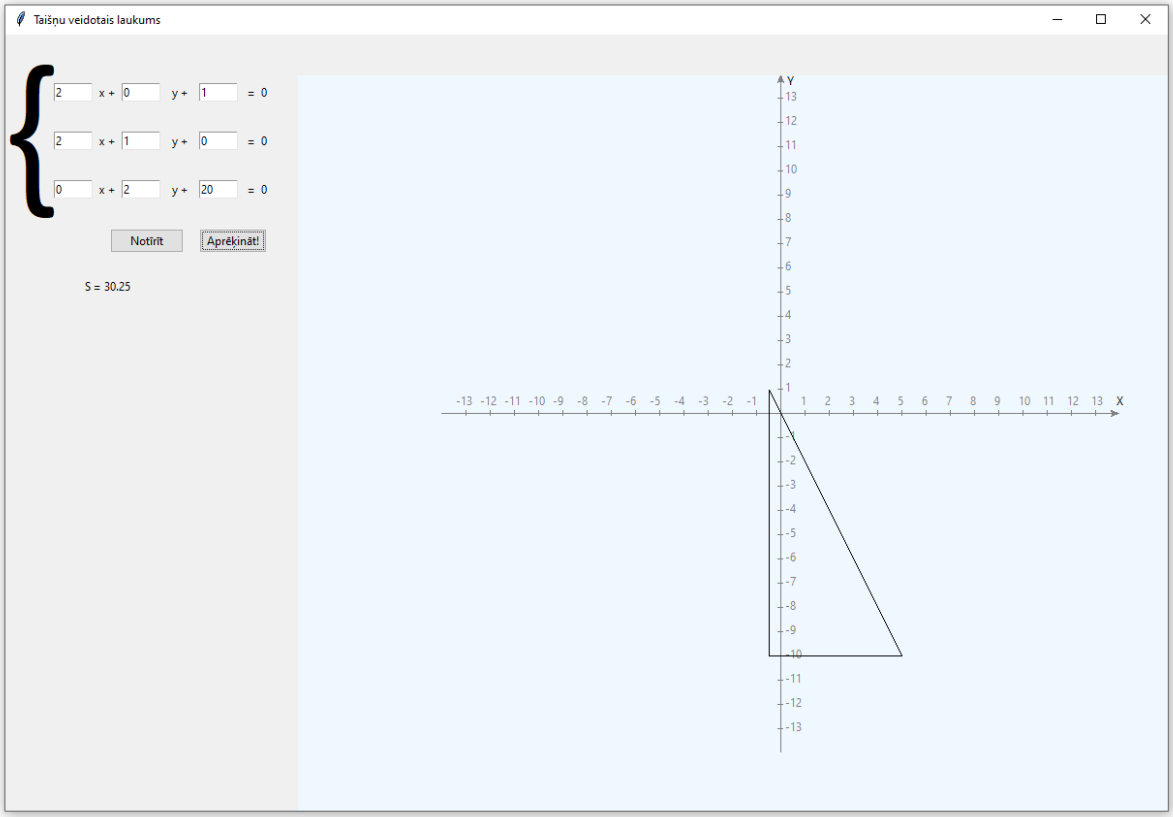
3)


Taišņu veidotais laukums
2 x + 2 y + 1 = 0
1 x + 2 y + 1.21 = 0
4 x + 2 y + 20 = 0
Notīrīt
Aprēķināt!
S = 15.714016666666666
Y
X

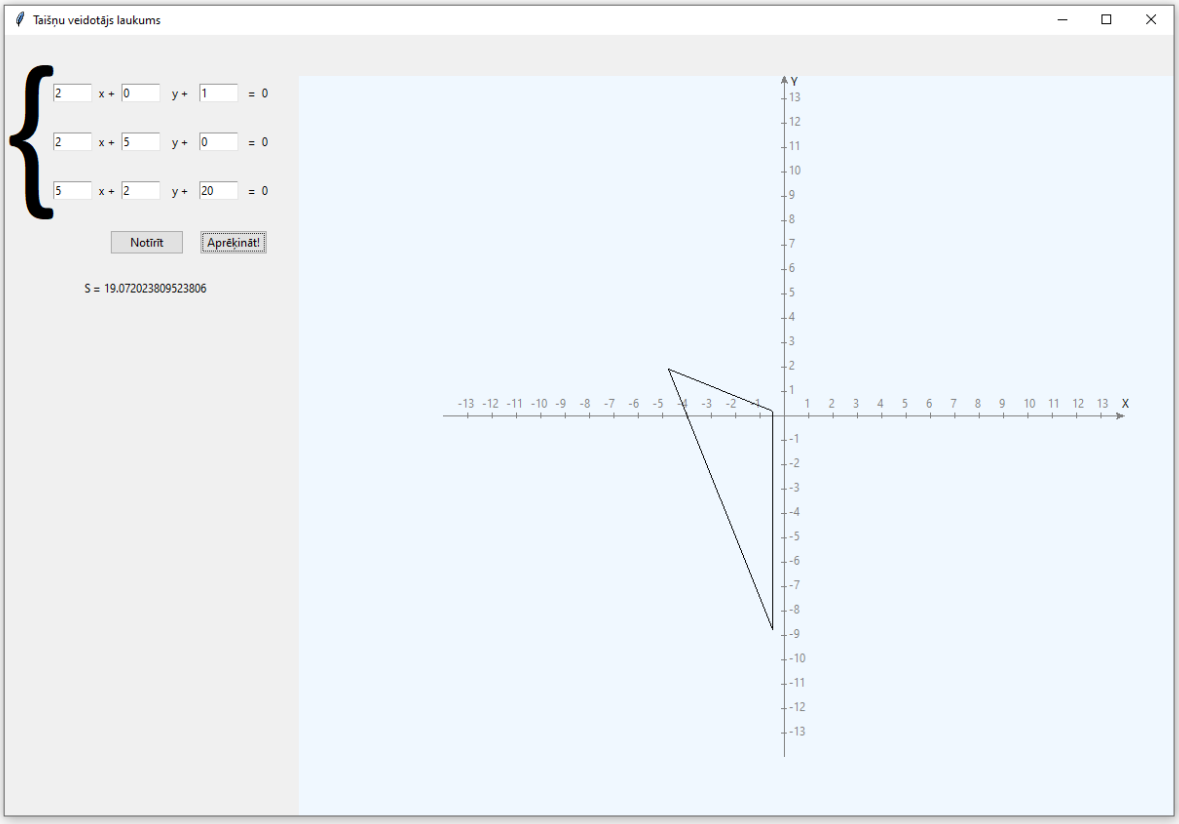
4)


Taišņu veidotais laukums
2 x + 2 y + 1 = 0
1 x + 2 y + 1.21 = 0
1 x + 5 y + 20 = 0
Notīrīt
Aprēķināt!
S = 11.56481666666667
Y
X

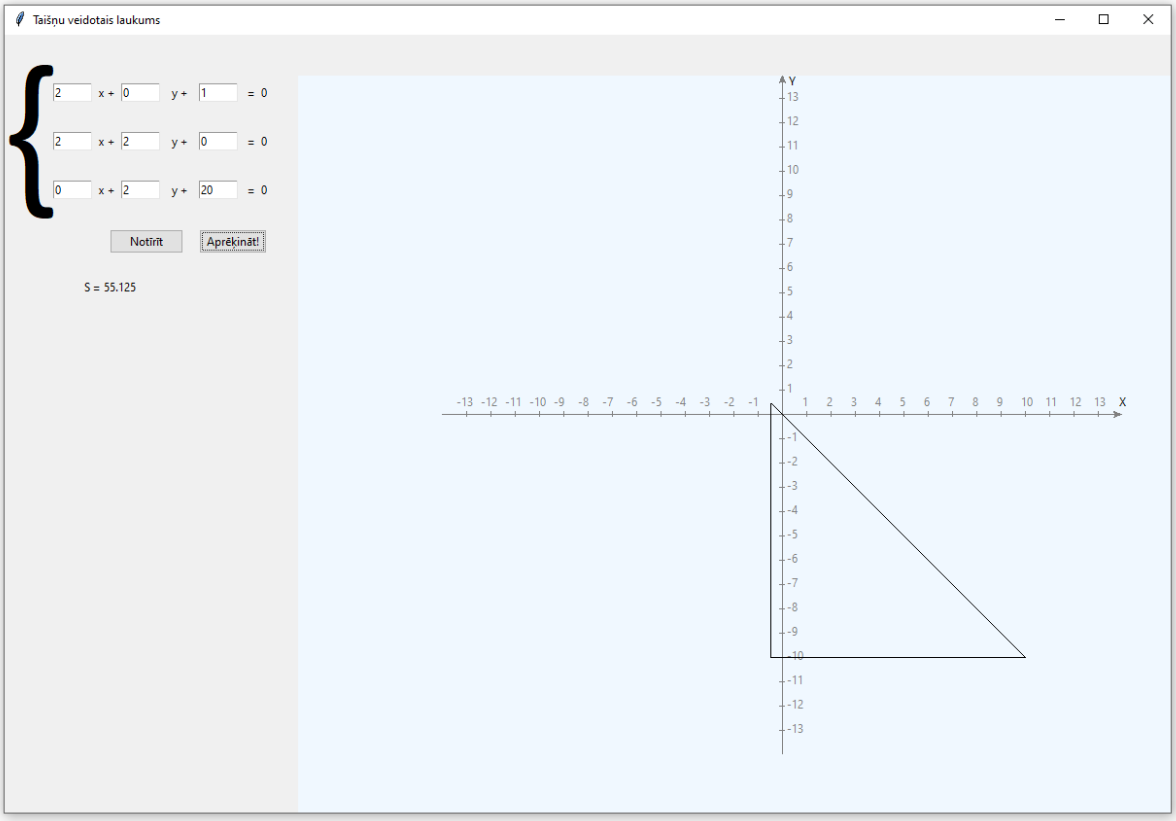
5)


Taišņu veidotais laukums
2 x + 0 y + 1 = 0
2 x + 1 y + 0 = 0
0 x + 2 y + 20 = 0
Notīrīt
Aprēķināt!
S = 30.25
Y
X

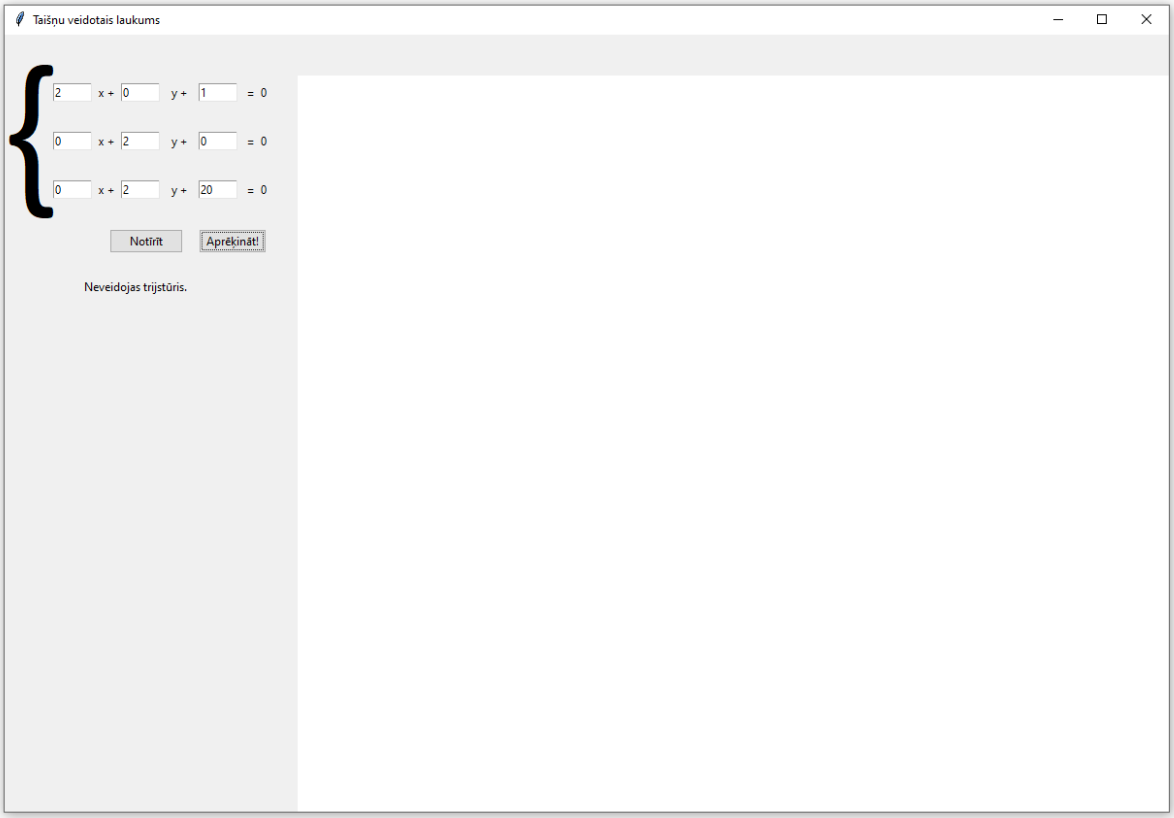
6)


Taišņu veidotājs laukums
2 x + 0 y + 1 = 0
2 x + 5 y + 0 = 0
5 x + 2 y + 20 = 0
Notīrīt
Aprēķināt!
S = 19.072023809523806
Y
X

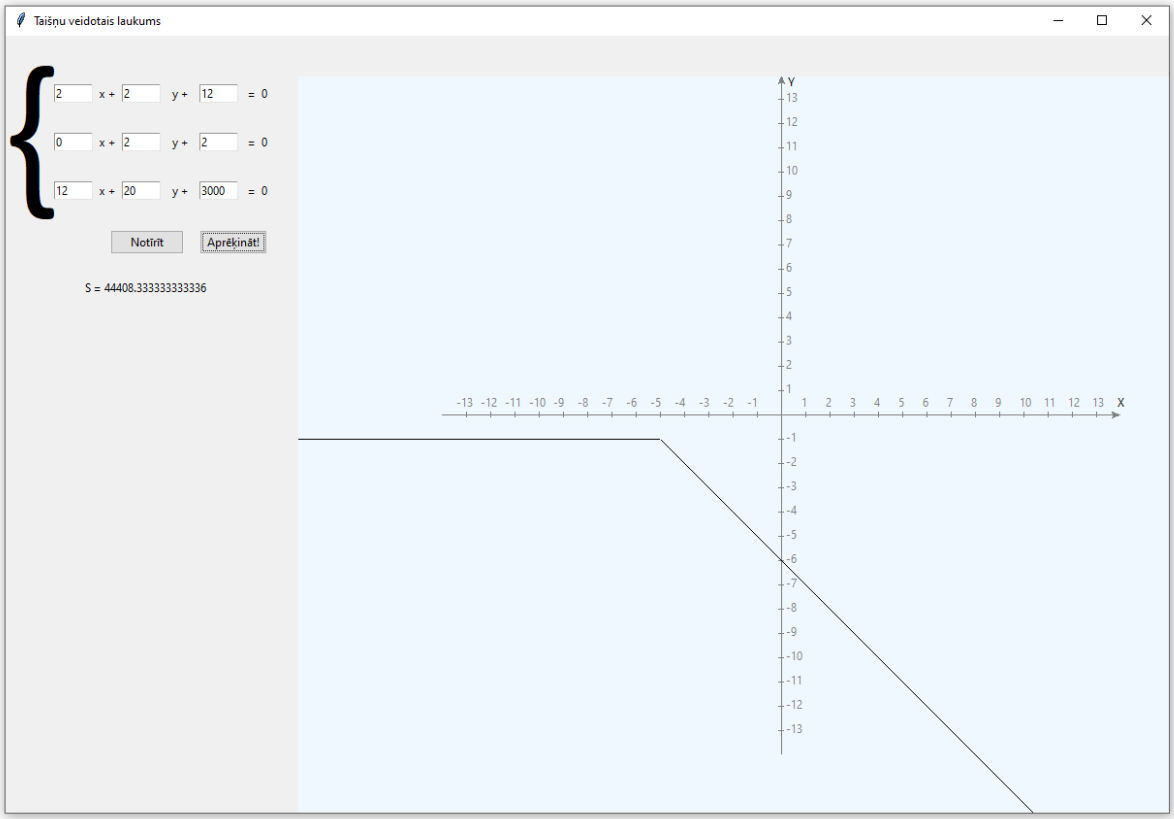
7)


Taišņu veidotais laukums
2 x + 0 y + 1 = 0
2 x + 2 y + 0 = 0
0 x + 2 y + 20 = 0
Notīrīt
Aprēķināt!
S = 55.125
Y
X

8)


Taišņu veidotais laukums
2 x + 0 y + 1 = 0
0 x + 2 y + 0 = 0
0 x + 2 y + 20 = 0
Notīrīt
Aprēķināt!
Neveidojas trijstūris.

9)


Taišņu veidotais laukums
2 x + 2 y + 12 = 0
0 x + 2 y + 2 = 0
12 x + 20 y + 3000 = 0
Notīrīt
Aprēķināt!
S = 44408.333333333336
Y
X

PU2. uzdevums

Sastādīt programmu, kas pārbauda vai ievadīta korekta e-pasta adrese. Simbolu virknes iebūvētās funkcijas izmantot nedrīkst, izņemot simbolu virknes caurskatīšanu pa vienam simbolam.

Pieņemsim, ka korektā e-pasta adresē:

1) jābūt tieši vienam @ simbolam

2) jābūt vismaz 1 rakstzīmei pirms un aiz simbola @ un punkta .

3) ir ne vairāk kā 256 rakstzīmes

4) ir jāsākas ar burtu vai ciparu

5) atsevišķie vārdi var saturēt tikai latīņu burtus, ciparus un pasvītrojuma rakstzīmi _

Kods:

```
# Programmas nosaukums: PU2 MPR15

# PU2 MPR15

# Uzdevuma formulējums:

'''

Sastādīt programmu, kas pārbauda vai ievadīta korekta e-pasta adrese. Simbolu virknes iebūvētās funkcijas izmantot nedrīkst, izņemot simbolu virknes caurskatīšanu pa vienam simbolam.

Pieņemsim, ka korektā e-pasta adresē:

1) jābut tieši vienam @ simbolam

2) jābūt vismaz 1 rakstzīmei pirms un aiz simbola @ un punkta .

3) ir ne vairāk kā 256 rakstzīmes

4) ir jāsākas ar burtu vai ciparu

5) atsevišķie vārdi var saturēt tikai latīņu burtus, ciparus un pasvītrojuma rakstzīmi _

'''

# Versija 1.0



def is_valid_email(email):

    # pārbauda, vai e-pasts ir īsāks par 256 rakstzīmēm

    if len(email) > 256:

        return False
```

```
symbol_at_count = 0 # @ simbolu skaits, (sākuma ir 0) ja != 1 tad False
dot_count = 0       # punktu . skaits. (Sākuma ir 0)

if email[0] == "_": # Jo ir jāsakas ar burtu vai ciparu (pārbaude uz visiem legāliem burtiem ir tālak)
   return False

for i in range(0,len(email)):
   if email[i] == "@"  :
      symbol_at_count += 1
      # pārbauda, vai ir vismaz 1 rakstzīme pirms simbola @
      if i == 0:
         return False
      # pārbauda, vai ir vismaz 1 rakstzīme pēc simbola @
      if i == len(email) - 1:
         return False

   elif email[i] == ".":
      dot_count += 1
      # pārbauda, vai ir vismaz 1 rakstzīme pirms punkta
      if i == 0:
         return False
      # pārbauda, vai ir vismaz 1 rakstzīme pēc punkta
      if i == len(email) - 1:
         return False

      if email[i+1] == ".": # pārbauda, pēc punkta nav otrā punkta, ja ir tad False
         return False

   elif not ( email[i] == "q" or email[i] == "w" or email[i] == "e" or email[i] == "r" or
         email[i] == "t" or email[i] == "y" or email[i] == "u" or email[i] == "i" or
```

```
            email[i] == "o" or email[i] == "p" or email[i] == "a" or email[i] == "s" or
            email[i] == "d" or email[i] == "f" or email[i] == "g" or email[i] == "h" or
            email[i] == "j" or email[i] == "k" or email[i] == "l" or email[i] == "z" or
            email[i] == "x" or email[i] == "c" or email[i] == "v" or email[i] == "b" or
            email[i] == "n" or email[i] == "m" or

            email[i] == "Q" or email[i] == "W" or email[i] == "E" or email[i] == "R" or
            email[i] == "T" or email[i] == "Y" or email[i] == "U" or email[i] == "I" or
            email[i] == "O" or email[i] == "P" or email[i] == "A" or email[i] == "S" or
            email[i] == "D" or email[i] == "F" or email[i] == "G" or email[i] == "H" or
            email[i] == "J" or email[i] == "K" or email[i] == "L" or email[i] == "Z" or
            email[i] == "X" or email[i] == "C" or email[i] == "V" or email[i] == "B" or
            email[i] == "N" or email[i] == "M" or

            email[i] == "1" or email[i] == "2" or email[i] == "3" or email[i] == "4" or
            email[i] == "5" or email[i] == "6" or email[i] == "7" or email[i] == "8" or
            email[i] == "9" or email[i] == "0" or

            email[i] == "_" or email[i] == "."):

        # atsevišķie vārdi var saturēt tikai latīņu burtus, ciparus un pasvītrojuma rakstzīmi _

        return False


if symbol_at_count != 1: # Jābut tieši vienam @ simbolam
    return False
if dot_count == 0: # jābut vismaz vienam punktam. @inbox.lv @lu.lv
    return False

return True
```

```
# ----------------- galvenā programma

email = input("Ievadiet derīgo e-pastu => ")

is_valid_email(email)
while True:
    if is_valid_email(email) == False:
        email = input("\nTas nav korekts e-pasts!\nIevadiet derīgo e-pastu => ")
        is_valid_email(email)
    else:
        print("Tas ir korekts e-pasts!")
        break
```

Testa piemēri:

1)

```
Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => @labiinbox.lv

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => .labi@inbox.lv

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => _labi@inbox.lv

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => _____labi@inbox.lv

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => labi@inbox.lv.

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => []!!!!/////@inbox.lv

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => labi..@inbox.lv

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => labi@

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => @

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => .

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => labi...@inbox.lv

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => labi.@inbox.lv
Tas ir korekts e-pasts!
```

2)

```
Ievadiet derīgo e-pastu => labi.labi.labi.labi.labi@lu.lu.lu.lu.lu.lu.lu
Tas ir korekts e-pasts!
```

3)

```
Ievadiet derīgo e-pastu => ЖЖЖ@INBOX.LV

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => Skudra@2.labi.lv
Tas ir korekts e-pasts!
```

4)

```
Ievadiet derīgo e-pastu => NoVeMbRa@InBoX.LvLvLlvLVLvlsaasasr1122r2asfasf
Tas ir korekts e-pasts!
```

5)

```
Ievadiet derīgo e-pastu => ZzZxCvVbBnmMjJ0gGuaygdAUaywqdADBa@U3333.LV.LV.LV.LV.LVFASFAS22222_________2.PG
Tas ir korekts e-pasts!
```

6)

```
Ievadiet derīgo e-pastu => das   as asfasf as as a @ @ 2 @ @ @2rrqq .  .as.as.fasf.asf.asf.asf..asf  .21.12

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => asd......

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => .....

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => @@@@@

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => @......@.

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => @rasdf@

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => @@..22

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => fasfasx355ззззз:''''''''''''''''''''''''''''''''''''

Tas nav korekts e-pasts!
Ievadiet derīgo e-pastu => vbabanins@edu.riga.lv
Tas ir korekts e-pasts!
```

7)

```
Ievadiet derīgo e-pastu => uff@ibbbbbb.lllllllllllllllllllllllllllllllllll.lllllllllllllllll.vb
Tas ir korekts e-pasts!
```