#2.daļa

# Eksāmens studiju kursā Programmēšana un datori II (2.daļa)

1. **uzdevums**

#1.uzdevums

def naudasGabali(n):

#funkcija veic nepieciešamo naudas gabalu aprēķinu #nosaka, cik banknotes nepieciešamas banknotes=[500,200,100,50,20,10,5]

v=int(n) #veselā daļa no ievadītās summas

for i in range(7):

while v >= banknotes[i]: a=v//banknotes[i] v=v-banknotes[i]\*a

print(a,"-",banknotes[i],"euro banknotes")

#nosaka cik euro monētas nepieciešamas

monetasEUR=[2,1] for i in range(2):

while v>=monetasEUR[i]: a=v//monetasEUR[i] v=v-monetasEUR[i]\*a

print(a,"-",monetasEUR[i],"euro monētas")

#nosaka cik centu monētas nepieciešamas

centi=[50,20,10,5,2,1]

l=int(n) #palīgmainīgais ar sākotnējās summas veselo daļu

b=round((n-l),2)

c=int(b\*100) #c- centu summa for i in range(6):

while c>=centi[i]: a=c//centi[i] c=c-centi[i]\*a

print(a,"-",centi[i],"centu monētas")

x=float(input("Ievadiet summu: ")) naudasGabali(x)

# uzdevums

#2.daļa

#2.uzdevums

#autors Diāna Antužāne

import numpy def ievade(n):

#funkcija realizē divdimensiju masīva ievadi

a = numpy.empty((n, n)) for i in range(n):

for j in range(n):

a[i,j] = int(input("ievadi matricas "

+ "elememtu ("

+ str(i) + ","

+ str(j) + ") ===>"))

return a def izvade(a):

#funkcija realizē divdimensiju masīva izvadi

n = a.shape[0] # x axis m = a.shape[1] # y axis s = ""

for i in range(n):

for j in range(m):

s = s + "{:5.0f}".format(a[i,j]) s = s + "\n"

return s

def rinduSumma(a):

#funkcija aprēķina matricas rindu summu

n=a.shape[0]

c=numpy.zeros(n) #rezultējošais masīvs

for i in range(n):

s=0

for j in range(n):

s=s+a[i,j] c[i]=s

return c

def kolonnuSumma(a):

#funkcija aprēķina matricas kolonnu summu

n=a.shape[0]

c=numpy.zeros(n) #rezultējošais masīvs

for i in range(n): s=0

for j in range(n): s=s+a[j,i]

c[i]=s return c

def diagonaluSumma(a):

#funkcija aprēķina matricas diagonāļu summu

n = a.shape[0]

c = numpy.zeros(n) #c ir rezultējošais masīvs

s=0

#galvenās diagonāles summa

for i in range(n):

s=s+a[i,i] c[0]=s

s=0

#blakus diagonāles summa

for i in range(n):

s=s+a[i,n-1-i] c[1]=s

#c[0] ir galvenās diagonāles elementu summa #c[1] ir blakus diagonāles elementu summa

return c

#galvenā programma

n=int(input("Ievadiet kvadrātitskas matricas izmēru==>"))

a=ievade(n) b=izvade(a)

print("Ievadītā matrica:")

print(b)

#izsauc summu funkcijas c=diagonaluSumma(a) r=rinduSumma(a) k=kolonnuSumma(a)

#rezultāta izvadīšanas formatēšana

#jauns masīvs, kas saturēs gan ievadīto matricu, gan visas summas malās

mat=numpy.zeros((n+2,n+2)) for i in range(n+2):

#galveno diagonāļu summas

if i==0:

mat[i,i]=c[0] elif i==n+1:

mat[i,0]=c[1]

mat[i,i]=c[0]

mat[0,i]=c[1]

#kolonnu un rindu summas else:

mat[0,i]=k[i-1]

mat[n+1,i]=k[i-1]

mat[i,0]=r[i-1]

mat[i,n+1]=r[i-1]

#sākotnējās matricas ievadīšana rezultējošās matricas centrā

for j in range(n+1):

if i>0 and j>0 and i<n+1: mat[i,j]=a[i-1,j-1]

print("Rindu un kolonnu summas") print(izvade(mat))

#2.daļa

#3.uzdevums

#autors Diāna Antužāne

import numpy

# uzdevums

def ievade(n):

#funkcija realizē divdimensiju masīva ievadi

a = numpy.empty((n, n)) for i in range(n):

for j in range(n):

if i==j or i>j:

#atlikušie elementi ir elementi uz diagonāles un elementi, kuru rindas indekss lielāks par

kolonnas indeksu

a[i,j] = input("ievadi matricas "

+ "elememtu ("

+ str(i) + ","

+ str(j) + ") ===>")

return a

def izvade(a):

#funkcija nodrošina matricas glītu izvadi

n = a.shape[0] # x axis m = a.shape[1] # y axis s = ""

for i in range(n):

for j in range(m):

s = s + "{:6.0f}".format(a[i,j]) s = s + "\n"

return s

def aizpilda(a):

#funkcija aizpilda matricas tukšās vietas

n = a.shape[0] m = a.shape[1]

b = numpy.empty((m,n))

for i in range(m):

for j in range(n):

if i<j:

#aizpilda tukšās vietas transponējot

b[i,j]=a[j,i] elif i>j or i==j:

#ieraksta ievadīto matricu

b[i,j]=a[i,j] return b

#galvenā programma

n=int(input("Ievadiet matricas izmēru"))

a=ievade(n)

print("Jānīša matrica pirms atjaunošanas")

print(izvade(a))

print("Jānīša matrica pēc atjaunošanas")

print(izvade(aizpilda(a)))

#2.daļa

#4.uzdevums

#autors Diāna Antužāne

import random

# uzdevums

def lodisuIzloze():

#funkcija veic sākotnējo piecu lodīšu izlozi

b=set()

for i in range(5):

l=random.randint(1,35) while l in b:

l=random.randint(1,35) b.add(l)

return b

def atkartotaIzloze(b):

#funkcija veic atkārtotu vienas lodītes lozēšanu un pievieno to izlozēto lodīšu kopai

l=random.randint(1,35) while l in b:

l=random.randint(1,35) b.add(l)

return b

def izvade(a):

#funkcija nodrošina datu glītu izvadi simbolu virknes formā

a=sorted(a) sv=""

for i in range(len(a)):

if i<len(a)-1:

sv=sv+str(a[i])+"," else:

sv=sv+str(a[i]) return sv

#galvenā programma

janisaL=lodisuIzloze() #jānīša izlozētās lodītes

peterisaL=lodisuIzloze() #pēterīša izlozētās lodītes

laimests=1024 meginajumi=1

x=janisaL.intersection(peterisaL) #kopīgās lodītes

#ja šķēluma kopas garums ir 5, tātad izlozētas vienādas lodītes

if len(x)==5: laimests=laimests meginajumi=meginajumi

else:

#ja šķēluma kopas garums nav pieci, tad lozē papildus lodītes

while len(x)<5: atkartotaIzloze(janisaL) atkartotaIzloze(peterisaL) x=janisaL.intersection(peterisaL)

else:

i=len(janisaL)-5 #aprēķina kāpinātāju laimesta aprēķināšanai

meginajumi=i+meginajumi laimests=laimests/2\*\*i

#rezultāta izvade

print("Tika izdarīti",meginajumi,"mēģinājumi." ) print("Jānītis izvilka lodītes -",izvade(janisaL))

print("Pēterītis izvilka lodītes -",izvade(peterisaL)) print("Kopīgās lodītes -",izvade(x)) print("Laimests - EUR","{:0.2f}".format(laimests))

# uzdevums

*#1 #2 #3 #4 #3 #4 #5 #5 #2 #4 #5 #3 #10 #6 #7 #7 #8 #5 #6 #8 #7 #8 #11 #10 #8 #8 #3 #11 #7*

|  |  |
| --- | --- |
| *Nr* | *Frāze* |
| *1* | *A* |
| *2* | *B* |
| *3* | *AB* |
| *4* | *BA* |
| *5* | *ABB* |
| *6* | *BAA* |
| *7* | *ABBA* |
| *8* | *AA* |
| *9* | *ABBAB* |
| *10* | *BB* |
| *11* | *BBB* |
| *12* | *BAAB* |

# uzdevums

#2.daļa #6.uzdevums #atver datnes

#datne1 no kuras nolasa datus un datne2, kurā ieraksta datus. datne1 = open("OGAS.txt", mode="r", encoding="utf-8") datne2 = open("POGAS.txt", mode="w", encoding="utf-8")

#no datne1 nolasa datus pa vienam simbolam simbols = datne1.readline(1)

while simbols != "": #kamēr simbols nav nekas

sk=0 #ievieš skaitītāju, kas uzskaita atstarpju skaitu

while simbols==" ": #kamēr simbols ir atstarpe, nolasa nākamo simbolu

sk=sk+1 simbols=datne1.readline(1)

else:

if sk>0:

#ja skaitītājs nav 0

#tad bijušas vairākas atstarpes, kuras tika dzēstas #nepieciešams pievienot atstarpi pirms simbola simbols=" "+simbols

datne2.write(simbols) simbols = datne1.readline(1)

datne1.close() datne2.close()