**14. praktiskais darbs. 2. semestris**

**1. uzdevums**

Sekmju žurnāla izveide. Vienas skolas visu skolēnu sekmju uzskaite glabājas vārdnīca, kurā katrs skolēns tiek identificēts pēc tā personas koda (11 ciparu virkne).

Par katru skolēnu tiek uzkrāta šāda informācija:

1. Vārds

2. Uzvārds

3. Klase

4. Mācību priekšmeti un tajos saņemtās atzīmes (mācību priekšmetu un tajos saņemto atzīmju skaitu katrai klasei nosaka programmas autors)

Pēc skolas izveides un atzīmju ievades jāparedz, ka tiek izveidots "baltais žurnāls", kur redzamas katra skolēna gada atzīmes katrā mācību priekšmetā, kā arī "liecības" izveidi, ierakstot informāciju teksta datnē ar nosaukumu VardsUzvards.txt

Testējot programmu, jāparedz, kā skolā ir vismaz 10 skolēni, kas mācās vismaz 3 dažādās klasēs un katrā klasē ir vismaz 3 mācību priekšmeti, kuros izliek vismaz 3 atzīmes. (5 punkti).

**Kods:**

# Programmas nosaukums: Skolas sekmju uzskaite

# 1. uzdevums (1MPR14\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sekmju žurnāla izveide. Vienas skolas visu skolēnu sekmju uzskaite glabājas vārdnīca, kurā katrs skolēns tiek identificēts pēc tā personas koda (11 ciparu virkne).

# Par katru skolēnu tiek uzkrāta šāda informācija:

# 1. Vārds

# 2. Uzvārds

# 3. Klase

# 4. Mācību priekšmeti un tajos saņemtās atzīmes (mācību priekšmetu un tajos saņemto atzīmju skaitu katrai klasei nosaka programmas autors)

# Pēc skolas izveides un atzīmju ievades jāparedz, ka tiek izveidots "baltais žurnāls", kur redzamas katra skolēna gada atzīmes katrā mācību priekšmetā, kā arī

# "liecības" izveidi, ierakstot informāciju teksta datnē ar nosaukumu VardsUzvards.txt

# Testējot programmu, jāparedz, kā skolā ir vismaz 10 skolēni, kas mācās vismaz 3 dažādās klasēs un katrā klasē ir vismaz 3 mācību priekšmeti, kuros izliek vismaz 3 atzīmes. (5 punkti).

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

"""

ĒRTĪBAS PĒC PROGRAMMA TIKA IZVEIDOTI DIVI FAILI

.CSV FAILS, KURU AIZPILD PATS LIETOTĀJS AR ROKU

.TXT FAILS, KUR TIEK VEIDOTAS LIECĪBAS UN KURU IZVEIDO PROGRAMMA

TIKA REALIZĒTS TIEŠI TĀ, JO TĀ IR ĒRTĀK LIETOTĀJAM, NEVISS IEVĀDĪT PROGRAMMA ĻOTI DAUDZ SKOLĒNU VĀRDU, UZVĀRDU, PERSONAS ID UN ATZĪMES.

TĀPĒC PROGRAMMA NAV PAREDZĒTA FUNCKIJA KAS PAPRASA LIETOTĀJAM IEVĀDĪT SĒKMES VAI SKOLĒNUS.

"""

import csv

def read\_student\_data(file\_name):

# Šī funkcija nolasa skolēna datus no CSV faila un sakārto tos vārdnīcā.

# file\_name - ceļš līdz failām (.csv datne).

# Piemēram:

# file\_name = "C:\\Users\\User\\Desktop\\student\_data.csv"

# Tas atgriež skolēna datu vārdnīcu.

# return student\_data

student\_data = {} # Definējam tukšu vārdnicu

with open(file\_name, 'r', newline='', encoding='utf-8-sig') as file:

# Datu izvilkšana no CSV rindām un kārtošana vārdnīcā

# Katrs skolēns tiek identificēts ar unikālu skolēna kodu

reader = csv.DictReader(file, delimiter=';')

for row in reader:

# Izvelcam attiecīgos datus no katras CSV faila rindas un sakārtojam tos vārdnīcā

student\_code = row['Personas kods'] # Izvelcam skolēna unikālo ID identifikatoru (Personas kodu) no kolonnas "Personas kods"

student\_name = row['Vārds'] # Izvelcam skolēna uzvārdu no rindas "Vārds".

student\_last\_name = row['Uzvārds'] # Izvelcam skolēna uzvārdu no rindas "Uzvārds".

student\_class = row['Klase'] # Izvelcam skolēna klasi no rindas "Klase".

math\_grades = [int(grade) for grade in row['Matemātika'].split()] # Izņemam matemātikas atzīmes no rindas "Matemātika" un pārveidojam "int" skaitļos

science\_grades = [int(grade) for grade in row['Dabaszinības'].split()] # Izņemam dabaszīnibas atzīmes no rindas "Dabaszinības" un pārveidojam "int" skaitļos

history\_grades = [int(grade) for grade in row['Vēsture'].split()] # Izņemam vēstures atzīmes no rindas "Vēsture" un pārveidojam "int" skaitļos

geo\_grades = [int(grade) for grade in row['Ģeogrāfija'].split()] # Izņemam ģeogrāfijas atzīmes no rindas "Ģeogrāfija" un pārveidojam "int" skaitļos

computer\_grades = [int(grade) for grade in row['Datorika'].split()] # Izņemam datorika atzīmes no rindas "Datorika" un pārveidojam "int" skaitļos

# Izveido vārdnīcu ar nosaukumu subject\_grades, lai saglabātu katra priekšmeta atzīmes. Vārdnīca ir strukturēta šādi:

# Priekšmets: Atzīmē priekšmeta nosaukumu ('Matemātika', 'Dabaszinības', 'Vēsture', 'Ģeogrāfija', 'Datorika').

# Atzīmes: atspoguļo attiecīgajam mācību priekšmetam atbilstošo atzīmju sarakstu.

subject\_grades = {

'Matemātika': math\_grades,

'Dabaszinības': science\_grades,

'Vēsture': history\_grades,

'Ģeogrāfija': geo\_grades,

'Datorika': computer\_grades

}

# Pievienojam skolēna informāciju un atzīmes vārdnīcai student\_data

student\_data[student\_code] = {'Vārds': student\_name, 'Uzvārds': student\_last\_name, 'Klase': student\_class, 'Marks': subject\_grades}

return student\_data

def print\_student\_grades\_by\_code(student\_data, student\_code):

# Šī funkcija izdrukā skolēna atzīmes, kas identificētas pēc skolēna koda.

# student\_data: vārdnīca, kurā ir skolēna dati.

# studenta\_kods: skolēna unikālais identifikators (Skolēna ID (Personas kods)).

# Atgriež: None

if student\_code in student\_data: # Pārbaudam, vai skolēna datu vārdnīcā ir šāds skolēna kods (vai tāds skolēns eksistē)

student\_info = student\_data[student\_code] # Iegūvam informāciju par doto skolēna kodu

student\_name = student\_info['Vārds'] # Iegūvam informāciju par doto skolēna vārdu

student\_surname = student\_info['Uzvārds'] # Iegūvam informāciju par doto skolēna uzvārdu

student\_grades = student\_info['Marks'] # Iegūvam informāciju par doto skolēna atzīmem

print(f"Atzīmes skolēnam: {student\_name} {student\_surname}") # Uzrakstām lietotājam skolēna atzīmes.

for subject, grades in student\_grades.items(): # Ejam ciklā caur katru priekšmetu un tām atbilstošam atzīmem un izdrūkam tos

print(f"Priekšmets: {subject}, Atzīmes: {grades}") # Izdrukājam priekšmeta nosaukumu un tā atzīmes

else: # Izdrukājam paziņojumu, kas norāda, ka skolēns ar norādīto kodu netika atrasts, ja netika atrāsts šāds personas kods.

print(f"Skolēns ar personas kodu '{student\_code}' nav atrāsts.")

def calculate\_average(grades):

# Aprēķina vidējo atzīmi no atzīmju saraksta.

# Tas atgriež vidējo atzīmi kā noapaļotu līdz veselam skaitlim.

# (Ja nevajag apoļot, tad return average)

# Ja atzīmju saraksts ir tukšs, tas atgriež 0.

total = sum(grades)

count = len(grades)

if count == 0:

return 0 # Gadījums, kad nav atzīmēs.

average = total / count

return round(average)

def generate\_white\_journal(student\_data):

# Ģenerē baltu žurnālu, pamatojoties uz skolēna datiem.

# Tas atgriež vārdnīcu, kurā ir priekšmeti kā atslēgas, un skolēnu vārdu un vidējo atzīmju vārdnīcu kā vērtības.

# Atgriež baltu žurnālu.

white\_journal = {} # Inicializējam tukšu vārdnīcu, lai saglabātu balto žurnālu.

for student\_name, data in student\_data.items(): # Iterējam caur katru skolēnu student\_datu vārdnīcā.

student\_class = data['Klase'] # Iegūvam skolēna klasi

subject\_grades = data['Marks'] # Iegūvam skolēna atzīmes

for subject, grades in subject\_grades.items(): # Iterējam caur katru priekšmetu un tā atbilstošās atzīmem.

average\_grade = calculate\_average(grades) # Aprēķinam katra priekšmeta vidējo atzīmi un saglabājam to baltajā žurnālā.

if subject not in white\_journal: # Ja priekšmets vēl nav iekļauts vārdnīcā white\_journal, pievienojam to.

white\_journal[subject] = {}

white\_journal[subject][student\_name] = average\_grade # Pievienojam skolēna vārdu un vidējo atzīmi attiecīgajam priekšmetam baltajā žurnālā.

return white\_journal # Atgriež baltu žurnālu.

def generate\_testimony(student\_data):

# Šī funkcija ģenerē liecību, pamatojoties uz skolēna datiem.

# Tas atgriež simbolu virkni, kas satur liecību.

# Ja skolēnam vidēja atzīme ir 0 (visas atzīmes ir 0), tad tas priekšmēts neparadīsies (vajag ierākstīt 0 priekšmētos kurus nav skolēnam).

testimony = ""

for student\_name, data in student\_data.items():

student\_class = data['Klase']

subject\_grades = data['Marks']

testimony = testimony + f"Liecība ID: {student\_name} (Klase: {student\_class}):\n"

for subject, grades in subject\_grades.items():

average\_grade = calculate\_average(grades)

if average\_grade != 0:

testimony = testimony + f"Priekšmets: {subject}, Vidējā atzīme: {average\_grade}\n"

testimony = testimony + "\n"

return testimony

def write\_testimony\_to\_file(testimony, file\_name):

# Ieraksta ģenerēto liecību .txt failā (file\_name)

# Tas neatgriež nekādu vērtību (atgriež None)

with open(file\_name, 'w', encoding='utf-8') as file:

file.write(testimony)

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

# Galvenā programmas daļa

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

file\_name = "C:\\Users\\User\\Desktop\\student\_data.csv" # Aizvietojiet ar sava CSV faila nosaukumu

student\_data = read\_student\_data(file\_name)

# TESTĒŠANAI ------------------------

# VAR PĀRBAUDĪT KATRU SKOLENU UN VIŅAS VISAS ATZĪMES

student\_code = '10000000000' # Aizstāt ar skolēna kodu, kuram vēlaties piekļūt

print\_student\_grades\_by\_code(student\_data, student\_code)

# TESTĒŠANAI ------------------------

# Izveidot un izdrukāt balto žurnālu

white\_journal = generate\_white\_journal(student\_data)

# Izveidot liecību un ierakstit to failā

testimony = generate\_testimony(student\_data)

file\_name = "C:\\Users\\User\\Desktop\\VardsUzvards.txt" # Aizvietojiet ar sava TXT faila nosaukumu

write\_testimony\_to\_file(testimony, file\_name)

print("\nLiecības tika uzģenerētas un ierakstītas failā VardsUzvards.txt") # Informācija lietotājam

**Testa piemēri:**

1)

A black screen with white text

Description automatically generated with low confidence

2)

Liecība ID: 10000000000 (Klase: 10A):

Priekšmets: Matemātika, Vidējā atzīme: 1

Priekšmets: Dabaszinības, Vidējā atzīme: 7

Priekšmets: Vēsture, Vidējā atzīme: 10

Priekšmets: Ģeogrāfija, Vidējā atzīme: 6

Priekšmets: Datorika, Vidējā atzīme: 4

Liecība ID: 10000000001 (Klase: 11B):

Priekšmets: Matemātika, Vidējā atzīme: 5

Priekšmets: Dabaszinības, Vidējā atzīme: 6

Priekšmets: Vēsture, Vidējā atzīme: 9

Priekšmets: Ģeogrāfija, Vidējā atzīme: 8

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000002 (Klase: 1A):

Priekšmets: Matemātika, Vidējā atzīme: 2

Priekšmets: Dabaszinības, Vidējā atzīme: 2

Priekšmets: Vēsture, Vidējā atzīme: 6

Liecība ID: 10000000003 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 5

Priekšmets: Vēsture, Vidējā atzīme: 6

Priekšmets: Datorika, Vidējā atzīme: 8

Liecība ID: 10000000004 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 7

Priekšmets: Vēsture, Vidējā atzīme: 7

Priekšmets: Datorika, Vidējā atzīme: 7

Liecība ID: 10000000005 (Klase: 1A):

Priekšmets: Matemātika, Vidējā atzīme: 1

Priekšmets: Dabaszinības, Vidējā atzīme: 2

Priekšmets: Vēsture, Vidējā atzīme: 2

Liecība ID: 10000000006 (Klase: 10A):

Priekšmets: Matemātika, Vidējā atzīme: 10

Priekšmets: Dabaszinības, Vidējā atzīme: 9

Priekšmets: Vēsture, Vidējā atzīme: 6

Priekšmets: Ģeogrāfija, Vidējā atzīme: 5

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000007 (Klase: 12A):

Priekšmets: Matemātika, Vidējā atzīme: 10

Priekšmets: Dabaszinības, Vidējā atzīme: 10

Priekšmets: Vēsture, Vidējā atzīme: 10

Priekšmets: Ģeogrāfija, Vidējā atzīme: 10

Priekšmets: Datorika, Vidējā atzīme: 10

Liecība ID: 10000000008 (Klase: 12A):

Priekšmets: Matemātika, Vidējā atzīme: 9

Priekšmets: Dabaszinības, Vidējā atzīme: 4

Priekšmets: Vēsture, Vidējā atzīme: 9

Priekšmets: Ģeogrāfija, Vidējā atzīme: 7

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000009 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 9

Priekšmets: Vēsture, Vidējā atzīme: 5

Priekšmets: Datorika, Vidējā atzīme: 9

3)

A picture containing text, document, font, screenshot

Description automatically generated

**PU1. uzdevums**

Papildināt 1.programmu, kas atļauj skolēnu atzīmju ievadi un rediģēšanu pa vienai atzīmei vienam skolēnam. (5 punkti).

**Kods:**

# Programmas nosaukums: Skolas sekmju uzskaite ar skolēnu atzīmju ievadi un rediģēšanu

# PU1. uzdevums (1MPR14\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Papildināt 1.programmu, kas atļauj skolēnu atzīmju ievadi un rediģēšanu pa vienai atzīmei vienam skolēnam. (5 punkti).

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

"""

Programma tika papildināta ar rediģēšanas iespēju.

Lietotājs ievada Personas ID un tiek parādīts visu atzīmju saraksts, lietotājas ievada noteikto priekšmetu un pēc tam viņš ievada jaunās atzīmes.

Atzīmes tika mainītas .csv failā un pēc tam tika izveidots .txt fails.

ĒRTĪBAS PĒC PROGRAMMA TIKA IZVEIDOTI DIVI FAILI

.CSV FAILS, KURU AIZPILD PATS LIETOTĀJS AR ROKU

.TXT FAILS, KUR TIEK VEIDOTAS LIECĪBAS UN KURU IZVEIDO PROGRAMMA

TIKA REALIZĒTS TIEŠI TĀ, JO TĀ IR ĒRTĀK LIETOTĀJAM, NEVISS IEVĀDĪT PROGRAMMA ĻOTI DAUDZ SKOLĒNU VĀRDU, UZVĀRDU, PERSONAS ID UN ATZĪMES.

TĀPĒC PROGRAMMA NAV PAREDZĒTA FUNCKIJA KAS PAPRASA LIETOTĀJAM IEVĀDĪT SĒKMES VAI SKOLĒNUS.

"""

import csv

def read\_student\_data(file\_name):

# Šī funkcija nolasa skolēna datus no CSV faila un sakārto tos vārdnīcā.

# file\_name - ceļš līdz failām (.csv datne).

# Piemēram:

# file\_name = "C:\\Users\\User\\Desktop\\student\_data.csv"

# Tas atgriež skolēna datu vārdnīcu.

# return student\_data

student\_data = {} # Definējam tukšu vārdnicu

with open(file\_name, 'r', newline='', encoding='utf-8-sig') as file:

# Datu izvilkšana no CSV rindām un kārtošana vārdnīcā

# Katrs skolēns tiek identificēts ar unikālu skolēna kodu

reader = csv.DictReader(file, delimiter=';')

for row in reader:

# Izvelcam attiecīgos datus no katras CSV faila rindas un sakārtojam tos vārdnīcā

student\_code = row['Personas kods'] # Izvelcam skolēna unikālo ID identifikatoru (Personas kodu) no kolonnas "Personas kods"

student\_name = row['Vārds'] # Izvelcam skolēna uzvārdu no rindas "Vārds".

student\_last\_name = row['Uzvārds'] # Izvelcam skolēna uzvārdu no rindas "Uzvārds".

student\_class = row['Klase'] # Izvelcam skolēna klasi no rindas "Klase".

math\_grades = [int(grade) for grade in row['Matemātika'].split()] # Izņemam matemātikas atzīmes no rindas "Matemātika" un pārveidojam "int" skaitļos

science\_grades = [int(grade) for grade in row['Dabaszinības'].split()] # Izņemam dabaszīnibas atzīmes no rindas "Dabaszinības" un pārveidojam "int" skaitļos

history\_grades = [int(grade) for grade in row['Vēsture'].split()] # Izņemam vēstures atzīmes no rindas "Vēsture" un pārveidojam "int" skaitļos

geo\_grades = [int(grade) for grade in row['Ģeogrāfija'].split()] # Izņemam ģeogrāfijas atzīmes no rindas "Ģeogrāfija" un pārveidojam "int" skaitļos

computer\_grades = [int(grade) for grade in row['Datorika'].split()] # Izņemam datorika atzīmes no rindas "Datorika" un pārveidojam "int" skaitļos

# Izveido vārdnīcu ar nosaukumu subject\_grades, lai saglabātu katra priekšmeta atzīmes. Vārdnīca ir strukturēta šādi:

# Priekšmets: Atzīmē priekšmeta nosaukumu ('Matemātika', 'Dabaszinības', 'Vēsture', 'Ģeogrāfija', 'Datorika').

# Atzīmes: atspoguļo attiecīgajam mācību priekšmetam atbilstošo atzīmju sarakstu.

subject\_grades = {

'Matemātika': math\_grades,

'Dabaszinības': science\_grades,

'Vēsture': history\_grades,

'Ģeogrāfija': geo\_grades,

'Datorika': computer\_grades

}

# Pievienojam skolēna informāciju un atzīmes vārdnīcai student\_data

student\_data[student\_code] = {'Vārds': student\_name, 'Uzvārds': student\_last\_name, 'Klase': student\_class, 'Marks': subject\_grades}

return student\_data

def print\_student\_grades\_by\_code(student\_data, student\_code):

# Šī funkcija izdrukā skolēna atzīmes, kas identificētas pēc skolēna koda.

# student\_data: vārdnīca, kurā ir skolēna dati.

# studenta\_kods: skolēna unikālais identifikators (Skolēna ID (Personas kods)).

# Atgriež: None

if student\_code in student\_data: # Pārbaudam, vai skolēna datu vārdnīcā ir šāds skolēna kods (vai tāds skolēns eksistē)

student\_info = student\_data[student\_code] # Iegūvam informāciju par doto skolēna kodu

student\_name = student\_info['Vārds'] # Iegūvam informāciju par doto skolēna vārdu

student\_surname = student\_info['Uzvārds'] # Iegūvam informāciju par doto skolēna uzvārdu

student\_grades = student\_info['Marks'] # Iegūvam informāciju par doto skolēna atzīmem

print(f"Atzīmes skolēnam: {student\_name} {student\_surname}") # Uzrakstām lietotājam skolēna atzīmes.

for subject, grades in student\_grades.items(): # Ejam ciklā caur katru priekšmetu un tām atbilstošam atzīmem un izdrūkam tos

print(f"Priekšmets: {subject}, Atzīmes: {grades}") # Izdrukājam priekšmeta nosaukumu un tā atzīmes

else: # Izdrukājam paziņojumu, kas norāda, ka skolēns ar norādīto kodu netika atrasts, ja netika atrāsts šāds personas kods.

print(f"Skolēns ar personas kodu '{student\_code}' nav atrāsts.")

def calculate\_average(grades):

# Aprēķina vidējo atzīmi no atzīmju saraksta.

# Tas atgriež vidējo atzīmi kā noapaļotu līdz veselam skaitļim.

# (Ja nevajag apoļot, tad return average)

# Ja atzīmju saraksts ir tukšs, tas atgriež 0.

total = sum(grades)

count = len(grades)

if count == 0:

return 0 # Gadījums, kad nav atzīmēs.

average = total / count

return round(average)

def generate\_white\_journal(student\_data):

# Ģenerē baltu žurnālu, pamatojoties uz studenta datiem.

# Tas atgriež vārdnīcu, kurā ir priekšmeti kā atslēgas, un skolēnu vārdu un vidējo atzīmju vārdnīcu kā vērtības.

# Atgriež baltu žurnālu.

white\_journal = {} # Inicializējam tukšu vārdnīcu, lai saglabātu balto žurnālu.

for student\_name, data in student\_data.items(): # Iterējam caur katru skolēnu student\_datu vārdnīcā.

student\_class = data['Klase'] # Iegūvam skolēna klasi

subject\_grades = data['Marks'] # Iegūvam skolēna atzīmes

for subject, grades in subject\_grades.items(): # Iterējam caur katru priekšmetu un tā atbilstošās atzīmem.

average\_grade = calculate\_average(grades) # Aprēķinam katra priekšmeta vidējo atzīmi un saglabājam to baltajā žurnālā.

if subject not in white\_journal: # Ja priekšmets vēl nav iekļauts vārdnīcā white\_journal, pievienojiet to.

white\_journal[subject] = {}

white\_journal[subject][student\_name] = average\_grade # Pievienojam skolēna vārdu un vidējo atzīmi attiecīgajam priekšmetam baltajā žurnālā.

return white\_journal # Atgriež baltu žurnālu.

def generate\_testimony(student\_data):

# Šī funkcija ģenerē liecību, pamatojoties uz skolēna datiem.

# Tas atgriež simbolu virkni, kas satur liecību.

# Ja skolēnam vidēja atzīme ir 0 (visas atzīmes ir 0), tad tas priekšmēts neparadīsies (vajag ierākstīt 0 priekšmētos kurus nav skolēnam).

testimony = ""

for student\_name, data in student\_data.items():

student\_class = data['Klase']

subject\_grades = data['Marks']

testimony = testimony + f"Liecība ID: {student\_name} (Klase: {student\_class}):\n"

for subject, grades in subject\_grades.items():

average\_grade = calculate\_average(grades)

if average\_grade != 0:

testimony = testimony + f"Priekšmets: {subject}, Vidējā atzīme: {average\_grade}\n"

testimony = testimony + "\n"

return testimony

def write\_testimony\_to\_file(testimony, file\_name):

# Ieraksta ģenerēto liecību .txt failā (file\_name)

# Tas neatgriež nekādu vērtību (atgriež None)

with open(file\_name, 'w', encoding='utf-8') as file:

file.write(testimony)

def change\_student\_marks(file\_name, student\_id, subject, new\_marks):

# Izmainā skolēna atzīmes noteiktā priekšmeta uz jaunajām.

# Atgriež True, ja netika atrāsta neviena kļūda.

# Atgriež False, ja tika atrāsta kļūda.

# file\_name - .csv failā ceļš

# student\_id - skolēna personas kods

# subject - skolas priekšmēts

# new\_marks - jaunās atzīmes

student\_data = read\_student\_data(file\_name) # Nolasa skolēnu datus no CSV faila

if student\_id in student\_data: # Pārbauda, vai skolēna ID ir iekšā student\_data

# Iegūst skolēna informāciju un atzīmes

student\_info = student\_data[student\_id]

student\_grades = student\_info['Marks']

# Pārbauda, vai priekšmets pastāv skolēna atzīmēs

if subject in student\_grades:

# Atjaunina atzīmes ar jaunajām atzīmēm

student\_grades[subject] = new\_marks

# Ieraksta atjauninātos skolēna datus atpakaļ CSV failā

with open(file\_name, 'w', newline='', encoding='utf-8-sig') as file:

kolonnas\_nosaukumi = ['Personas kods', 'Vārds', 'Uzvārds', 'Klase', 'Matemātika', 'Dabaszinības', 'Vēsture', 'Ģeogrāfija', 'Datorika']

writer = csv.DictWriter(file, fieldnames=kolonnas\_nosaukumi, delimiter=';')

writer.writeheader()

for student\_code, student\_info in student\_data.items():

# Ieraksta skolēna informāciju un atzīmes CSV failā

writer.writerow({

'Personas kods': student\_code, # Ierakstam skolēna unikālo identifikatoru kolonnā "Personas kods".

'Vārds': student\_info['Vārds'], # Ailē 'Vārds' ierakstiet skolēna vārdu

'Uzvārds': student\_info['Uzvārds'], # Ailē 'Uzvārds' ierakstiet skolēna uzvārdu

'Klase': student\_info['Klase'], # Ierakstam skolēna klasi kolonnā "Klase".

'Matemātika': ' '.join(str(grade) for grade in student\_info['Marks']['Matemātika']), # Konvertējam un savienojam visas matemātikas atzīmes ar atstarpi atdalītā simbolu virknē un ierakstisim to rindiņa "Matemātika"

'Dabaszinības': ' '.join(str(grade) for grade in student\_info['Marks']['Dabaszinības']), # Konvertējam un savienojam visas dabaszīnibas atzīmes ar atstarpi atdalītā simbolu virknē un ierakstisim to rindiņa "Dabaszinības"

'Vēsture': ' '.join(str(grade) for grade in student\_info['Marks']['Vēsture']), # Konvertējam un savienojam visas vēstures atzīmes ar atstarpi atdalītā simbolu virknē un ierakstisim to rindiņa "Vēsture"

'Ģeogrāfija': ' '.join(str(grade) for grade in student\_info['Marks']['Ģeogrāfija']), # Konvertējam un savienojam visas ģeogrāfijas atzīmes ar atstarpi atdalītā simbolu virknē un ierakstisim to rindiņa "Ģeogrāfija"

'Datorika': ' '.join(str(grade) for grade in student\_info['Marks']['Datorika']) # Konvertējam un savienojam visas datorikas atzīmes ar atstarpi atdalītā simbolu virknē un ierakstisim to rindiņa "Datorika"

})

return True # Atgriež True, lai norādītu uz veiksmīgu skolēna atzīmju izmainīšanu

else:

print(f"Skolēns ar personas kodu '{student\_id}' nav atrasts priekšmetā '{subject}'.")

else:

print(f"Skolēns ar personas kodu '{student\_id}' nav atrasts.")

return False # Atgriež False, lai norādītu uz neveiksmigu skolēnu atzīmju izmainīšanu

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

# Galvenā programmas daļa

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

file\_name = "C:\\Users\\User\\Desktop\\student\_data.csv"

student\_data = read\_student\_data(file\_name)

student\_code = ""

while student\_code.lower() != "nē": # Turpinām ciklu, līdz ievade ir "nē".

# Lūdzam lietotājam ievadīt skolēna personas kodu vai "mē", lai izietu no cikla

student\_code = input("Lūdzu, ievadiet skolēna personas kodu, kuras atzīmes gribāt izmainīt, ja negribat, tad ierakstiet \"Nē\" ==> ")

if student\_code.lower() != "nē" and student\_code in student\_data: # Ja ievade nav "Nē" (neatkarīgi no reģistra) un tāds skolēns eksistē, tad...

print\_student\_grades\_by\_code(student\_data, student\_code) # ... izsaucām funkciju, lai izdrukātu skolēna atzīmes, pamatojoties uz norādīto Personas kodu

subject = input("Lūdzu, ievadiet priekšmetu: ") # Lūdziet lietotājam skolas priekšmētu

if subject in student\_data[student\_code]['Marks']: # Ja priekšmets pastāv skolēna "atzīmju ierakstos"

# Lūdzam lietotājam ievadīt jaunās atzīmes, atdalot tās ar atstarpēm

new\_marks = input("Lūdzu, ievadiet jaunās atzīmes atdalītas ar atstarpi: ")

# Pārvēršam atzīmju ievades virkni par veselu skaitļu sarakstu

new\_marks = [int(grade) for grade in new\_marks.split()]

if change\_student\_marks(file\_name, student\_code, subject, new\_marks): # Izsaucām funkciju, lai mainītu skolēna atzīmes norādītajā priekšmetā

pass

else:

print("Notika neparedzēta kļūda!") # Testēšanai

else:

print(f"Skolēnam ar personas kodu '{student\_code}' nav atzīmes par priekšmētu '{subject}'.") # Ja netika atrāsts

else:

if student\_code.lower() == "nē":

pass

else:

print("Kļūda! Tāds skolēns neeksistē!")

break

file\_name = "C:\\Users\\User\\Desktop\\student\_data.csv" # Aizvietojiet ar sava CSV faila nosaukumu

student\_data = read\_student\_data(file\_name)

# Ģenerējam un izdrukājam balto žurnālu

white\_journal = generate\_white\_journal(student\_data)

# Izveidojam liecību un ierakstiet to failā

testimony = generate\_testimony(student\_data)

file\_name = "C:\\Users\\User\\Desktop\\VardsUzvards.txt" # Aizvietojiet ar sava TXT faila nosaukumu

write\_testimony\_to\_file(testimony, file\_name)

print("\nLiecības tika uzģenerētas un ierakstītas failā VardsUzvards.txt")

**Testa piemēri:**

1)

A picture containing text, screenshot, font

Description automatically generated

Liecība ID: 10000000000 (Klase: 10A):

Priekšmets: Matemātika, Vidējā atzīme: 8

Priekšmets: Dabaszinības, Vidējā atzīme: 7

Priekšmets: Vēsture, Vidējā atzīme: 10

Priekšmets: Ģeogrāfija, Vidējā atzīme: 6

Priekšmets: Datorika, Vidējā atzīme: 4

Liecība ID: 10000000001 (Klase: 11B):

Priekšmets: Matemātika, Vidējā atzīme: 5

Priekšmets: Dabaszinības, Vidējā atzīme: 6

Priekšmets: Vēsture, Vidējā atzīme: 9

Priekšmets: Ģeogrāfija, Vidējā atzīme: 8

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000002 (Klase: 1A):

Priekšmets: Matemātika, Vidējā atzīme: 2

Priekšmets: Dabaszinības, Vidējā atzīme: 2

Priekšmets: Vēsture, Vidējā atzīme: 6

Liecība ID: 10000000003 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 5

Priekšmets: Vēsture, Vidējā atzīme: 6

Priekšmets: Datorika, Vidējā atzīme: 8

Liecība ID: 10000000004 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 7

Priekšmets: Vēsture, Vidējā atzīme: 7

Priekšmets: Datorika, Vidējā atzīme: 7

Liecība ID: 10000000005 (Klase: 1A):

Priekšmets: Matemātika, Vidējā atzīme: 1

Priekšmets: Dabaszinības, Vidējā atzīme: 2

Priekšmets: Vēsture, Vidējā atzīme: 2

Liecība ID: 10000000006 (Klase: 10A):

Priekšmets: Matemātika, Vidējā atzīme: 10

Priekšmets: Dabaszinības, Vidējā atzīme: 9

Priekšmets: Vēsture, Vidējā atzīme: 6

Priekšmets: Ģeogrāfija, Vidējā atzīme: 5

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000007 (Klase: 12A):

Priekšmets: Matemātika, Vidējā atzīme: 10

Priekšmets: Dabaszinības, Vidējā atzīme: 10

Priekšmets: Vēsture, Vidējā atzīme: 10

Priekšmets: Ģeogrāfija, Vidējā atzīme: 10

Priekšmets: Datorika, Vidējā atzīme: 10

Liecība ID: 10000000008 (Klase: 12A):

Priekšmets: Matemātika, Vidējā atzīme: 9

Priekšmets: Dabaszinības, Vidējā atzīme: 4

Priekšmets: Vēsture, Vidējā atzīme: 9

Priekšmets: Ģeogrāfija, Vidējā atzīme: 7

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000009 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 9

Priekšmets: Vēsture, Vidējā atzīme: 5

Priekšmets: Datorika, Vidējā atzīme: 9

A picture containing text, font, screenshot

Description automatically generated

2)

A picture containing text, screenshot, font

Description automatically generated

(izmaiņas netiek paradītas uzreiz lietotājam, jo fails mainīsies kad tiek uzrakstīts “nē”)

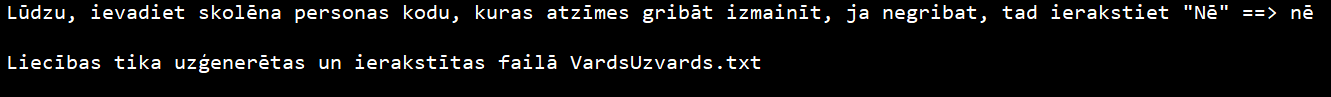
A picture containing text, font, screenshot

Description automatically generated

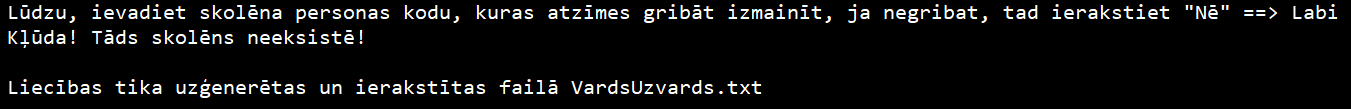
A black text on a white background

Description automatically generated with low confidence

3)



4)



5)

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, font, screenshot

Description automatically generated

Liecība ID: 10000000000 (Klase: 10A):

Priekšmets: Matemātika, Vidējā atzīme: 1

Priekšmets: Dabaszinības, Vidējā atzīme: 7

Priekšmets: Vēsture, Vidējā atzīme: 10

Priekšmets: Ģeogrāfija, Vidējā atzīme: 6

Priekšmets: Datorika, Vidējā atzīme: 10

Liecība ID: 10000000001 (Klase: 11B):

Priekšmets: Matemātika, Vidējā atzīme: 5

Priekšmets: Dabaszinības, Vidējā atzīme: 6

Priekšmets: Vēsture, Vidējā atzīme: 9

Priekšmets: Ģeogrāfija, Vidējā atzīme: 8

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000002 (Klase: 1A):

Priekšmets: Matemātika, Vidējā atzīme: 2

Priekšmets: Dabaszinības, Vidējā atzīme: 2

Priekšmets: Vēsture, Vidējā atzīme: 6

Liecība ID: 10000000003 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 5

Priekšmets: Vēsture, Vidējā atzīme: 6

Priekšmets: Datorika, Vidējā atzīme: 8

Liecība ID: 10000000004 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 7

Priekšmets: Vēsture, Vidējā atzīme: 7

Priekšmets: Datorika, Vidējā atzīme: 7

Liecība ID: 10000000005 (Klase: 1A):

Priekšmets: Matemātika, Vidējā atzīme: 1

Priekšmets: Dabaszinības, Vidējā atzīme: 2

Priekšmets: Vēsture, Vidējā atzīme: 2

Liecība ID: 10000000006 (Klase: 10A):

Priekšmets: Matemātika, Vidējā atzīme: 10

Priekšmets: Dabaszinības, Vidējā atzīme: 9

Priekšmets: Vēsture, Vidējā atzīme: 6

Priekšmets: Ģeogrāfija, Vidējā atzīme: 5

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000007 (Klase: 12A):

Priekšmets: Matemātika, Vidējā atzīme: 10

Priekšmets: Dabaszinības, Vidējā atzīme: 10

Priekšmets: Vēsture, Vidējā atzīme: 10

Priekšmets: Ģeogrāfija, Vidējā atzīme: 10

Priekšmets: Datorika, Vidējā atzīme: 10

Liecība ID: 10000000008 (Klase: 12A):

Priekšmets: Matemātika, Vidējā atzīme: 9

Priekšmets: Dabaszinības, Vidējā atzīme: 4

Priekšmets: Vēsture, Vidējā atzīme: 9

Priekšmets: Ģeogrāfija, Vidējā atzīme: 7

Priekšmets: Datorika, Vidējā atzīme: 5

Liecība ID: 10000000009 (Klase: 6A):

Priekšmets: Matemātika, Vidējā atzīme: 4

Priekšmets: Dabaszinības, Vidējā atzīme: 9

Priekšmets: Vēsture, Vidējā atzīme: 5

Priekšmets: Datorika, Vidējā atzīme: 10

**2. uzdevums**

Sastādīt programmu, kas nodrošina informācijas ievadi teksta datnē un nolasa informāciju no teksta datnes un parāda to uz ekrāna, ievērojot teksta sadalījumu pa rindkopām.

**Kods:**

# Programmas nosaukums: Datnes ievade un nolasīšana

# 2. uzdevums (1MPR14\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas nodrošina informācijas ievadi teksta datnē un nolasa informāciju no teksta datnes

# un parāda to uz ekrāna, ievērojot teksta sadalījumu pa rindkopām.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

def input\_lines\_by\_user():

# Metode, kas paprasā lietotājam ievādīt jaunu tekstu pa rindiņam un atgriež katru rindu, kā saraksta elementu.

# (Prasa ievadit i + 1 rindu, lai 0.rindu būtu uzrakstīta kā 1.rinda).

# Kad tiek uzrakstīts "...", tad apstāšana.

# Var izmantot datnes nolasīšanai un apstrādei.

# Atgriež sarakstu, kurā katrs elements ir rindas teksts (0.elements ir 0.rindas teksts, 1.elements ir 1.rindas teksts utt.)

lines = []

while True:

ievade = input(f"Ievadiet {len(lines) + 1}. rindas tekstu. Ja vēlies pabeigt, ievadi \"...\": ")

if ievade == "...":

break

lines.append(ievade)

return lines

def write\_to\_new\_file\_list(datne, lines):

# DZĒŠ VISU SATURU NO FAILĀ un ierāksta failā "datne" (piemēram, .txt failā) lines saraksts, kur katrs saraksta elements ir rindas teksts.

# lines - saraksts, kur katrs saraksta elements ir rindas teksts.

# datne - datnes fails (.txt)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

with open(datne, mode="w", encoding="utf-8") as datne:

for line in lines:

datne.write(line + "\n")

def print\_text\_from\_data\_by\_rows(datne):

# Uzrakstā termināla lietotājam visu tekstu no .txt failā pa rindam.

# datne - datnes fails (piemēram, .txt fails)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

print(rinda, end='')

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

# Galvenā programmas daļa

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

lines = input\_lines\_by\_user() # Izsaucam metodi, kas paprasa lietotājam ievādīt jaunu tekstu pa rindiņam.

datne = "C:\\Users\\User\\Desktop\\teksts.txt" # datne - ceļš līdz failām.

write\_to\_new\_file\_list(datne, lines) # NODZĒS VISU TEKSTU DATNĒ. Uzrakstīsim "datne" faila "lines" saraksta elementus, kur katru saraksta elementu izvadīsim kā atsevišķu rindiņu.

print("\nIerakstītais teksts datnē:") # Ši rinda netiek ierakstīta failā, tas ir tikai informācija lietotājam.

print\_text\_from\_data\_by\_rows(datne) # Izvadām lietotājam visu tekstu no .txt failā pa rindkopam.

**Testa piemēri:**

1)A black screen with white text

Description automatically generated with low confidence

2)

A black screen with white text

Description automatically generated with low confidence

3)

A screen shot of a computer

Description automatically generated with medium confidence

4)

A screenshot of a computer

Description automatically generated

**3. uzdevums**

Sastādīt programmu, kas nodrošina informācijas papildināšanu teksta datnē un parāda jauno datnes saturu uz ekrāna, ievērojot teksta sadalījumu pa rindkopām.

**Kods:**

# Programmas nosaukums: Datnes papildināšana un jaunas informācijas paradīšana

# 3. uzdevums (1MPR14\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas nodrošina informācijas papildināšanu teksta datnē un parāda jauno datnes saturu uz ekrāna,

# ievērojot teksta sadalījumu pa rindkopām.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

def input\_lines\_by\_user():

# Metode, kas paprasā lietotājam ievādīt jaunu tekstu pa rindiņam un atgriež katru rindu, kā saraksta elementu.

# (Prasa ievadit i + 1 rindu, lai 0.rindu būtu uzrakstīta kā 1.rinda).

# Kad tiek uzrakstīts "...", tad apstāšana.

# Var izmantot datnes nolasīšanai un apstrādei.

# Atgriež sarakstu, kurā katrs elements ir rindas teksts (0.elements ir 0.rindas teksts, 1.elements ir 1.rindas teksts utt.)

lines = []

while True:

ievade = input(f"Ievadi {len(lines) + 1}. rindas tekstu. Ja vēlies pabeigt, ievadi \"...\": ")

if ievade == "...":

break

lines.append(ievade)

return lines

def write\_to\_file\_list(lines, datne):

# Ierāksta datne failā (piemēram, .txt failā) lines saraksts, kur katrs saraksta elements ir rindas teksts.

# lines - saraksts, kur katrs saraksta elements ir rindas teksts.

# datne - datnes fails (.txt)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

with open(datne, mode="a", encoding="utf-8") as datne:

for line in lines:

datne.write(line + "\n")

def print\_text\_from\_data\_by\_rows(datne):

# Uzrakstā termināla lietotājam visu tekstu no .txt failā pa rindam.

# datne - datnes fails (piemēram, .txt fails)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

print(rinda, end='')

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

# Galvenā programmas daļa

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

datne = "C:\\Users\\User\\Desktop\\teksts.txt" # Ceļš līdz failām.

print("Ierakstītais teksts datnē:") # Informācija lietotājam.

print\_text\_from\_data\_by\_rows(datne) # Parādam lietotājam, kas tagad ir ierākstīts failā.

print() # Lai būtu atstārpe. (Glīti).

lines = input\_lines\_by\_user() # Izsaucam metodi, kas paprasa lietotājam ievādīt jaunu tekstu pa rindiņam.

write\_to\_file\_list(lines, datne) # Papildināsim "datne" failu ar "lines" sarakstu, kur katru saraksta elementu izvadīsim kā atsevišķu rindiņu.

print("\nPapildināts teksts datnē:") # Informācija lietotājam.

print\_text\_from\_data\_by\_rows(datne) # Parādam lietotājam, kas tagad ir ierākstīts failā pēc tas papildināšanas.

**Testa piemēri:**

1)

A screen shot of a computer

Description automatically generated with medium confidence

2)  
A computer screen with white text

Description automatically generated with low confidence

3)

A screen shot of a computer

Description automatically generated with medium confidence

**4. uzdevums**

Sastādīt programmu, kas nodrošina informācijas pārrakstīšanu (pārkopēšanu) no vienas teksta datnes otrā jaunā teksta datnē.

**Kods:**

# Programmas nosaukums: Vienas datnes informāciju pārkopēšanu otrā jaunā teksta datnē

# 4. uzdevums (1MPR14\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas nodrošina informācijas pārrakstīšanu (pārkopēšanu) no vienas teksta datnes otrā jaunā teksta datnē.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

def tekstu\_parkopesena\_no\_datne1\_to\_datne2(datne1, datne2):

# Pārkope tekstu no datne1 -> datne2.

# Atgriež None

# datne1 - no kurā failā gribām paņemt informāciju.

# datne2 - kurā failā gribām informāciju no datne1 pārkopēt.

# Piemēram:

# datne1 = "C:\\Users\\User\\Desktop\\teksts.txt"

# datne2 = "C:\\Users\\User\\Desktop\\teksts2.txt"

with open(datne1, mode="r", encoding="utf-8") as datne1:

with open(datne2, mode="w", encoding="utf-8") as datne2:

for rinda in datne1:

datne2.write(rinda)

def print\_text\_from\_data\_by\_rows(datne):

# Uzrakstā termināla lietotājam visu tekstu no .txt failā pa rindam.

# datne - datnes fails (piemēram, .txt fails)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

print(rinda, end='')

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

# Galvenā programmas daļa

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

# Ceļi līdz failiem

datne1 = "C:\\Users\\User\\Desktop\\teksts.txt"

datne2 = "C:\\Users\\User\\Desktop\\teksts2.txt"

print("Ierakstītais teksts 1.datnē:") # Informācija lietotājam.

print\_text\_from\_data\_by\_rows(datne1) # Parādam lietotājam kāda informācija ir ierākstīta failā datne1 (no kurā kopēsim informāciju).

print("\n") # Lai būtu glīti lietotājam.

print("Ierakstītais teksts 2.datnē:") # Informācija lietotājam.

print\_text\_from\_data\_by\_rows(datne2) # Parādam lietotājam kāda informācija ir ierākstīta failā datne2 (uz viņu mēs kopēsim informāciju).

print("\n") # Lai būtu glīti lietotājam.

tekstu\_parkopesena\_no\_datne1\_to\_datne2(datne1, datne2) # Veicām pārkopēšanu.

print("Ierakstītais teksts 2.datnē pēc pārrakstīšanas:") # Informācija lietotājam.

print\_text\_from\_data\_by\_rows(datne2) # Parādam lietotājam kāda informācija ir ierākstīta failā datne2 pēc pārkopēšanas (jābut informācijai no datne1).

**Testa piemēri:**

1)

A screenshot of a computer

Description automatically generated with medium confidence

2)

A black screen with white text

Description automatically generated with low confidence

3)

A screen shot of a computer

Description automatically generated with medium confidence

4)

A screenshot of a computer screen

Description automatically generated with medium confidence

5)

A screen shot of a computer

Description automatically generated with low confidence