תרגיל בית 4

הנחיות כלליות:

- קראו בעיון את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל הפתרונות לשאלות יחד בקובץ ex4_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז. שלכם, כל 9 הספרות כולל ספרת ביקורת.
- מועד אחרון להגשה: כמפורסם באתר.
- בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של התוכניות לקלטים שגויים, בכל שאלה הריצו את תוכניתכם עם מגוון קלטים שונים, אלה שהופיעו כדוגמאות בתרגיל וקלטים נוספים עליהם חשבתם (וודאו כי הפלט נכון).
- אין לשנות את שמות הפונקציות והמשתנים שכבר מופיעים בקובץ השלד של התרגיל.
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (עד לרמת הרווח).**
- אופן ביצוע התרגיל: שימו לב, בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.

שאלה 1

בשאלה זו נבנה מילון לקידוד מחרוזת בבסיס בינארי (מעל א"ב $\{0,1\}$) למחרוזת בבסיס עשרוני מתתי מחרוזות.

מספר b באורך n ספרות בבסיס בינארי (הבסיס מסומן באמצעות $[\ ]_2$) מתורגם לבסיס עשרוני כך: $\sum_{i=0}^{n-1} b[i] \times 2^i$, כך שהאינדקסים נמנים **מימין לשמאל***. לדוגמא, נראה את האינדקסים עבור המספר 11010 באורך 5:

ולכן:

$$\begin{array}{cccccc} & 4 & 3 & 2 & 1 & 0 \\ b = & 1 & 1 & 0 & 1 & 0 \end{array}$$

$$[11010]_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 26$$

נשים לב שהאפסים משמאל אינם משפיעים על ערך המספר, לדוגמא:

$$[0011]_2 = 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = [011]_2 = [11]_2 = 3$$

*הסבר מפורט על המרת מספר בבסיס בינארי לעשרוני ולהפך: https://en.wikipedia.org/wiki/Binary_number

א. כתוב פונקציה map_bin2dec אשר בהינתן מספר שלם וחיובי n מחזירה מילון, שהמפתחות בו הם <u>כל המחרוזות הבינאריות באורך n</u> והערכים הם <u>ייצוג מחרוזת</u> של המספר העשרוני התואם.
למשל, עבור $n = 1$, כל המחרוזות הבינאריות באורך 1 הן 0, 1 אשר מייצגות את המספרים 0, 1 בהתאמה. עבור $n = 2$, כל המחרוזות הבינאריות באורך 2 הן $00, 01, 10, 11$, אשר מייצגות את המספרים 0 עד 3 בהתאמה. בשני המקרים האלה הפונקציות יחזירו את המילונים כמודגם:

```
>>> map_bin2dec(1)
{'1': '1', '0': '0'}
>>> map_bin2dec(2)
{'11': '3', '10': '2', '00': '0', '01': '1'}
```

ניתן להניח ש-n הוא מספר שלם וחיובי ($n \geq 1$).

<u>טיפ</u>: בדומה למעברים בין טיפוסים שונים שראינו בהרצאות ובתרגולים int(), float(), str(), קיימת גם פונקציה למעבר ממספר שלם לבינארי bin(). הפונקציה הנ"ל מחזירה מחרוזת של הייצוג הבינארי עם הקידומת '0b' משמאל למספר. לדוגמא:

```
>>> bin(2)
'0b10'
>>> bin(0)
'0b0'
>>> bin(5)
'0b101'
```

לכן, כדי לקבל את המחרוזת הבינארית באורך n, נצטרך תחילה לקצץ שני תווים משמאל (את '0b') ואז למלא באפסים משמאל כך שבסוף הפעולה המחרוזת תהיה באורך n. פונקצית הספריה של פייתון zfill מאפשרת לעשות זאת. לדוגמא, אם ברצוננו לייצר מחרוזות בינאריות באורך 3 של המספרים 2, 0, 5 בדומה לדוגמאות לעיל, נעשה:

```
>>> bin(2)[2:].zfill(3)
'010'
>>> bin(0)[2:].zfill(3)
'000'
>>> bin(5)[2:].zfill(3)
'101'
```

ב. כתוב פונקציה bin_triplets_to_decimal שמקבלת מחרוזת בינארית bin_str מעל א"ב $\{0,1\}$, ומחזירה מחרוזת מעל א"ב $[0,7]$, כך שכל שלושה תווים ממחרוזת הקלט מתורגמים לספרה העשרונית התואמת. יש להשתמש לשם כך בפונקציה מסעיף א'.
התרגום יעשה במעבר משמאל לימין על כל שלשת תווים עוקבים שאינה חופפת. לדוגמא,

$$\underbrace{001}_{1}\ \underbrace{111}_{7}$$

ולכן עבור "001111" מחרוזת הפלט תהא "17". ניתן להניח שמחרוזת הקלט באורך שמתחלק ב-3 ללא שארית.
דוגמאות הרצה:

```
>>> bin_triplets_to_decimal("001111")
'17'
>>> bin_triplets_to_decimal("110011")
'63'
```

שאלה 2

א. נתון מבנה נתונים אשר מפרט את רשימת הקורסים של כל סטודנט. ברצוננו לייצר רשימה של סטודנטים לכל קורס.
כתוב פונקציה swap_student_courses שמקבלת מילון אשר המפתחות בו הם שמות של סטודנטים, ולכל סטודנט הערך הוא רשימה של מחרוזות אשר מייצגות את הקורסים שהסטודנט לומד. הפונקציה תחזיר מילון שהמפתחות בו הינם שמות הקורסים וערכיהם הם רשימות הסטודנטים בקורס.

שימו לב, אין חשיבות לסדר ברשימות בקלט ובפלט. הניחו כי שמות הקורסים והסטודנטים הינם case-sensitive, כלומר הקורס "Math" שונה מהקורס "math".
למשל,

```
>>> students_d = {"Yuval": ["Math", "Computer Science",
"Statistics"], "Gal": ["Algebra", "Statistics", "Physics"],
"Noam": ["Statistics", "Math", "Programming"]}

>>> swap_student_courses(students_d)
{'Statistics': ['Gal', 'Noam', 'Yuval'], 'Algebra': ['Gal'],
'Programming': ['Noam'], 'Computer Science': ['Yuval'],
'Physics': ['Gal'], 'Math': ['Noam', 'Yuval']}
```

ב. מנהל בית הספר רצה להבין את היחסים בין הקורסים השונים. הוא מעוניין לדעת עבור כל זוג קורסים כמה סטודנטים נרשמו לשניהם. בדוגמא לעיל, עבור הקורסים Math ו-Statistics יש שני סטודנטים משותפים (Noam ו-Yuval), ולקורסים Math ו-Physics ישנם 0 סטודנטים משותפים.
כתוב פונקציה count_courses_intersection אשר מקבלת מילון שמפתחותיו הם שמות הקורסים והערך של כל קורס הוא רשימה של סטודנטים (כמו מילון הפלט של סעיף א'). הפונקציה תחזיר מילון שהמפתחות שלו הינם זוגות של קורסים ולכל זוג קורסים, הערך יהיה מספר שלם המתאר את מספר הסטודנטים שלוקחים את שני הקורסים.

(המשך בעמוד הבא)

כפי שלמדנו, מפתחות של מילון חייבים להיות immutable, ולכן נשמור את זוגות הקורסים בתור tuples. אין חשיבות לסדר הקורסים בתוך כל זוג, אך כל זוג מופיע פעם אחת בלבד.

דוגמת הרצה:

```
>>> count_courses_intersection({'Bible': ['Daniel', 'Roni'],
'Psychology': ['Roni'], 'Biology': ['Daniel']})
{('Biology', 'Psychology'): 0, ('Biology', 'Bible'): 1,
('Bible', 'Psychology'): 1}
```

שאלה 3

משחק המילים Word chain הוא משחק בו כל שחקן בתורו זורק מילה אשר מתחילה באות בה המילה הקודמת הסתיימה. במשחק זה אסור לשחקן להשתמש במילה שכבר נאמרה מתחילת המשחק.

א. כתוב פונקציה find_following_words שמקבלת מחרוזת של אוסף מילים ומחזירה מילון כך שלכל מילה (מפתח) הערך הוא רשימה של כל המילים אשר מתחילות באות בה מסתיימת מילת המפתח. כל המילים במילון (מפתחות וערכים) יופיעו באותיות קטנות. רשימת המילים עבור כל מפתח צריכה להיות ממויינת בסדר לקסיקוגרפי (מילוני). מילה שחוזרת פעמיים בטקסט לא תופיע באותה רשימה במילון פעמיים. עבור מילה שאין לה עוקבת, הערך ברשימה יהיה רשימה ריקה. ניתן להניח שבמחרוזת הקלט כל שתי מילים מופרדות ברווח בודד, ושאין תווים מלבד אותיות ורווחים.
דוגמת הרצה:

```
>>> find_following_words("The little girl sings a Song")
{'a': ['a'], 'sings': ['sings', 'song'], 'little': [], 'song':
['girl'], 'the': [], 'girl': ['little']}
```

ב. כעת נדמה משחק של Word chain אשר מוגבל למילים בטקסט הנתון. כתוב פונקציה play_word_chain אשר מקבלת מחרוזת text כפי שמתואר בסעיף א', ומחרוזת נוספת word שהיא מילה אשר מופיעה ב-text (זכרו כי הטקסט והמילה הינם case-**in**sensitive, המילה יכולה להופיע בטקסט באותיות גדולות או קטנות). הפונקציה תתחיל ב-word ותדפיס למסך את המילה הבאה במשחק. המילה העוקבת לכל מילה היא הראשונה שמופיעה ברשימת המילים במילון וטרם היה שימוש בה במשחק. המילה האחרונה תהיה זו שאין לה עוקבות, או שכולן היו בשימוש. כל מילה תודפס בשורה חדשה באותיות קטנות. הפונקציה תחזיר רשימה של מילים שלא נעשה בהן שימוש. אין חשיבות לסדר ברשימה.

ניתן להניח ש-word קיימת בתוך text בתור מילה שלמה.

(המשך בעמוד הבא)

לפניכם שתי דוגמאות הרצה. מכיוון שלפונקציה יש ערך החזרה, בדוגמא אנו שומרים אותו למשתנה unused_words. בדוגמא השנייה למשל, למילה the לא קיימת עוקבת, כלומר מילה שמתחילה ב-e, ולכן the תודפס, אך כל שאר המילים מוחזרות ונשמרות לתוך unused_words:

```
>>> unused_words = play_word_chain("The little girl sings a
Song", "sings")
sings
song
girl
little
>>> unused_words
['a', 'the']

>>> unused_words = play_word_chain("The little girl sings a
Song", "tHe")
the
>>> unused_words
['a', 'girl', 'little', 'sings', 'song']
```

בהצלחה!