אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

תרגיל בית מספר 2 - להגשה עד 27/06/2024 בשעה 23:59

קראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

הגשה :

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- השתמשו בקובץ השלד skeleton2.py כבסיס לקובץ ה-py אותו אתם מגישים.
לא לשכוח לשנות את שם הקובץ למספר ת״ז שלכם לפני ההגשה, עם סיומת py.
- בסה״כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת״ז שלה הוא 012345678 הקבצים שיש להגיש הם hw2_012345678.pdf ו- hw2_012345678.py.
- הקפידו לענות על כל מה שנשאלתם.
- לפני ההגשה ודאו כי הרצתם את הפונקציה `test()` שבקובץ השלד אך זכרו כי היא מבצעת בדיקות בסיסיות בלבד וכי בתהליך הבדיקה הקוד ייבדק על פני מקרים מגוונים ומורכבים יותר.
שימו לב כי יש למחוק את **הקריאה** לפונקציה לפני ההגשה.

- בכל שאלה, אלא אם מצוין אחרת באופן מפורש, ניתן להניח כי הקלט תקין.

- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
להנחיה זו מטרה כפולה :
    1. על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
    2. כדי להרגיל אתכם להבעת טיעונים באופן מתומצת ויעיל, ללא פרטים חסרים מצד אחד אך ללא עודף בלתי הכרחי מצד שני. זוהי פרקטיקה חשובה במדעי המחשב.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

שאלה 1

עבור מספר שלם חיובי n נגדיר את $\boldsymbol{s(n)}$ להיות סכום כל המחלקים של n, לא כולל n עצמו.

לדוגמה, המחלקים של 4 הם $\{1,2\}$, ולכן $s(4) = 1 + 2 = 3$ (שימו לב ש-2 נספר פעם אחת).

מספר טבעי n נקרא **מספר משוכלל** (Perfect Number) אם $s(n) = n$. לדוגמה, 6 הוא מספר משוכלל כי:

$$s(6) = 1 + 2 + 3 = 6$$

סעיף א'

ממשו את הפונקציה $divisors(n)$ המחזירה רשימה של כל המחלקים של n, לא כולל n, בסדר עולה.
הנחיה מחייבת: יש לממש את הפונקציה באמצעות List Comprehension.

דוגמת הרצה:

```
>>> divisors(6)
[1, 2, 3]
>>> divisors(7)
[1]
```

סעיף ב'

ממשו את הפונקציה `perfect_numbers(c)` המחזירה רשימה של מספרים שלמים באורך c, הרשימה תכיל את c המספרים **המשוכללים** הראשונים (לפי סדר הופעתם). **כתבו בקובץ ה-PDF** את זמני הריצה עבור $c = 2, 3$. מה קורה עבור $c = 5$?
דוגמת הרצה:

```
>>> perfect_numbers(1)
[6]
>>> perfect_numbers(2)
[6, 28]
```

סעיף ג'

מספר טבעי n נקרא **מספר שופע** (Abundant Number) אם $s(n) > n$. לדוגמה, 12 הוא מספר שופע כי:

$$s(12) = 1 + 2 + 3 + 4 + 6 = 16 > 12$$

ממשו את הפונקציה $abundant\_density(n)$ אשר מחשבת את צפיפות המספרים השופעים מ-1 עד n, כלומר את היחס:

$$\frac{|\{k \in \mathbb{N} \mid k \leq n \text{ } and \text{ } k \text{ } is \text{ } abundant\}|}{n}$$

הפלט צריך להיות מספר מטיפוס `float` בקטע [0,1].

דוגמת הרצה:

```
>>> abundant_density(20) # 12, 18, 20 are abundant numbers
0.15
```

סעיף ד'

ידוע כי צפיפות המספרים השופעים, כש-n שואף לאינסוף, היא בין 0.2474 ל-0.2480 (צפיפותם המדויקת היא שאלה פתוחה). על מנת להשתכנע בנכונות הטענה, הריצו את הפונקציה עבור הערכים

$$n = 50, \quad 500, \quad 2500, \quad 5000, \quad 7500, \quad 10000$$

וכתבו את התוצאות בקובץ ה-PDF. סכמו בקצרה את הממצאים.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

סעיף ה'

מספר שלם חיובי n נקרא **מספר דמוי משוכלל** (Semi-perfect number) אם הוא שווה לסכום של **כל או חלק** מהמחלקים שלו (לא כולל n עצמו, ומבלי לסכום את אותו הגורם יותר מפעם אחת). למשל, המספר 18 הוא מספר דמוי משוכלל: המחלקים של 18 הם [1, 2, 3, 6, 9], ואכן

$$18 = 3 + 6 + 9$$

נאמר שמספר הוא **דמוי משוכלל מסדר k** אם הוא שווה לסכום של בדיוק k מן המחלקים שלו.
לדוגמה, 18 הוא מספר דמוי משוכלל מסדר 3.

ממשו את הפונקציה $semi\_perfect\_4(n)$ אשר מקבלת מספר n ומחזירה True אם המספר הוא דמוי משוכלל מסדר 4, ואחרת מחזירה False.

דוגמת הרצה:

```
>>> semi_perfect_4(20) # 20 = 1 + 4 + 5 + 10
True
>>> semi_perfect_4(28)
False
```

שימו לב ש-20 הוא מספר דמוי משוכלל מסדר 4 שאינו משוכלל, ו-28 הוא מספר משוכלל שאינו דמוי משוכלל מסדר 4.

סעיף ו'
(ללא קשר לסעיפים הקודמים)

בשאלה הבאה נממש גרסה איטרטיבית של אלגוריתם אוקלידס, למציאת מכנה משותף מקסימלי (Greatest Common Divisor, GCD).

Given with $a, b \in \mathbb{N}$
1. $t := b$
2. $b := a \,(mod\, b)$
3. $a := t$
4. If $b \neq 0$ go back to step **1**
5. return a as the GCD

מכנה משותף מקסימלי של שני מספרים טבעיים **מוגדר** להיות המחלק הגדול ביותר של שני המספרים. למשל, עבור המספרים 24 ו-54 קיימים המחלקים הבאים –

- המחלקים של 24 הם 1,2,3,4,6,8,12,24
- המחלקים של 54 הם 1,2,3,6,9,18,27,54

כלומר המחלקים שהם חולקים הם – 1,2,3,6 ; על כן, המחלק המשותף המקסימלי במקרה זה הוא **6**.

נתון פסאודו-קוד של האלגוריתם, עליכם לממש את הפונקציה `find_gcd(a, b)` שמקבלת שני מספרים טבעיים ומחזירה את ה-GCD שלהם.

שאלה 2

בשאלה זו נממש מספר פונקציות אקראיות תוך שימוש בפונקציה הבסיסית $random.random$,
ללא שימוש בפונקציות אחרות מהספרייה $random$.

כזכור, הספרייה $random$ מכילה פונקציה בשם $random$ שמחזירה מספר מטיפוס float בקטע [0,1), כאשר לכל מספר יש סיכוי שווה להיבחר (ליתר דיוק, לכל מספר שפייתון יודע לייצג בקטע [0,1) יש סיכוי שווה להיבחר). לצורך הפשטות נניח לאורך השאלה שעבור כל מספר $0 \leq p \leq 1$, הסיכוי לקבל מספר קטן מ-p בקטע [0,1) על ידי הפונקציה $random.random$ הוא p.

```
>>> import random
>>> random.random()
0.13937543523525686
>>> random.random()
0.6376812941041776
```

סעיף א'

ממשו את הפונקציה $coin()$ שמחזירה True בסיכוי חצי ו-False בסיכוי חצי.

סעיף ב'

ממשו את הפונקציה `sample(v, p)`. הפונקציה מקבלת רשימה של ערכים v באורך d, ורשימה p, גם באורך d, שמהווה התפלגות–כלומר מתקיים כי כל איבר ברשימה תחום בין 0 ל-1 וגם כי סכום איברי הרשימה הוא 1 ; בכתיב מתמטי :

$$\sum_{i=0}^{d-1} p[i] = 1 \wedge \forall i: p[i] \in [0,1]$$

על הפונקציה לדגום ולהחזיר איבר $v[i]$ בהסתברות של $p[i]$. ניתן להניח כי הקלט תקין וכי שתי הרשימות אינן ריקות.

```
>>> sample([5, 3, "s"],[0.1, 0.2, 0.7])
3  # with probability. 0.2

>>> sample([5, 3, "s"],[0.1, 0.2, 0.7])
"s"  # with probability 0.7
```

<u>הצעה</u>: על מנת לוודא שהפונקציה שלכם מגרילה כמצופה, ניתן לקחת השראה מדוגמא מההרצאה - בה הרצנו פונקציה אקראית מספר גדול של פעמים ובדקנו את התפלגות הפלטים שלה (אילו ערכים נצפה לשערך בשאלה?).

סעיף ג'

בסעיף זה נממש את בעיית מונטי-הול המפורסמת. הבעיה מבוססת על שעשועון המורכב משחקן, מנחה, 3 דלתות, 2 סלעים, ומכונית אחת – היא הפרס השווה. מהלך המשחק :

1. המנחה בוחר באקראי דלת **אחת** מתוך ה-3 ומציב מאחוריה את **המכונית** (מבלי שהשחקן יודע באיזה דלת בחר המנחה). מאחורי כל אחת משתי הדלתות האחרות הוא מציב סלע.
2. לאחר מכן השחקן נכנס לחדר, ובוחר דלת **אקראית אחת**, מאחוריה הוא מהמר שיש מכונית.
3. כעת המנחה פותח (וחושף) דלת אחת באקראי מבין הדלתות שהשחקן **לא בחר** ושמאחוריה יש **סלע** (יתכן שיש רק דלת אחת כזו ואז המנחה יבחר אותה).
4. בשלב זה נשארו רק שתי דלתות סגורות, ולשחקן ניתנת האפשרות לבחור האם ברצונו **להחליף** את הדלת שבחר בשלב 2 (לדלת הסגורה השנייה שנותרה) **או לא**.
5. בסוף, המנחה חושף את הדלת שהשחקן בחר, רק אם מסתתרת מאחוריה מכונית השחקן זוכה בפרס השווה.

נרצה להעריך מה האסטרטגיה האידאלית לשחקן : האם בשלב 4 כדאי לו להחליף את הדלת או לא? לשם כך ממשו את הפונקציה $monty\_hall(switch, times)$ אשר מקבלת משתנה בוליאני $switch$ ומספר שלם חיובי $times$. על

הפונקציה לבצע סימולציה של המשחק המתואר למעלה במשך $times$ פעמים, כאשר בשלב 4 השחקן בוחר אם לשנות את הדלת או לא לפי הערך של $switch$ (כלומר אם $switch$ שווה $True$ השחקן ישנה את הדלת הנבחרת בשלב 4, ואחרת הוא לא ישנה את הדלת שבחר). בסוף, הפונקציה מחזירה את סיכויי ההצלחה של האסטרטגיה $switch$ (כלומר את כמות הסימולציות בהן השחקן זכה במכונית, חלקי $times$). לדוגמא: אם $times = 10$ והשחקן זכה במכונית ב-5 סימולציות, הפונקציה תחזיר 0.5. דוגמת הרצה:

```
>>> monty_hall(True, 10)
0.5
```

נסו להריץ את הפונקציה עם הפרמטרים $monty\_hall(True, 10000)$, $monty\_hall(False, 10000)$. כתבו בקובץ ה-pdf את אחוזי ההצלחה שיצאו לכם עבור כל אחת מהאסטרטגיות. האם יש אסטרטגיה עדיפה לשחקן? (רשות: נסו להסביר למה זו התוצאה המתקבלת. בקורס בהסתברות בוודאי תחזרו לדוגמה הזו ותוכיחו את התוצאה הידועה).

סעיף ד'

השתמשו בסעיף ב' על מנת לממש את הפונקציה `sample_anagram(st)`, אשר מקבלת מחרוזת `st` ומחזירה אנגרמה אקראית שלה (במילים אחרות, פרמוטציה אקראית של רצף התווים הנתון), כאשר כל אנגרמה עשויה להיות מוחזרת בסיכוי שווה.

לדוגמא, עבור המחרוזת `st='abc'` הפונקציה תחזיר את אחת משש המחרוזות הבאות בסיכוי שווה:

`'abc', 'acb', 'bac', 'bca', 'cba', 'cab'`

דוגמת הרצה:

```
>>> sample_anagram("abcde")
"abcde"  # or any other anagram with the same probability…

>>> sample_anagram("abcde")
"cadeb"

>>> sample_anagram("basipacrachromatin")
"marsipobranchiata"
```

הנחיה מחייבת: בסעיף זה יש לקרוא לפונקציה $sample(v, p)$ שמימשתם בסעיף ב', **ואסור לקרוא ישירות** לפונקציה $random.random$ (או לכל פונקציה אחרת מספרייה חיצונית). $sample(v, p)$ יכולה לקרוא ל-$random.random$.

תזכורת ממטלה 1: אנגרמה היא מחרוזת שנוצרה מסידור מחדש של תווי מחרוזת אחרת (ללא הוספה/מחיקת אף תו). לדוגמא המחרוזת *silent* היא אנגרמה של המחרוזת *listen* (וההפך). בפרט, כל מחרוזת היא אנגרמה (טריוויאלית) של עצמה, וכל היפוך מחרוזת הוא אנגרמה של המחרוזת המקורית.

סעיף ה'

וודאו את נכונות הפונקציה שמימשתם בסעיף ד': הריצו את הפונקציה 10,000 פעמים על הקלט `st='abc'`, ופרטו בטבלה בקובץ ה-PDF כמה פעמים כל אחת מ-6 המחרוזות האפשריות התקבלה כפלט של הפונקציה. לאור התוצאות שצורפו, האם המימוש שלכם תקין? ציינו זאת ואת הסיבה לכך.

הערה: כתבו קוד בפייתון שמבצע את הבדיקה. וודאו כי מחקתם את הקוד הרלוונטי לסעיף זה לפני ההגשה.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2023-4

שאלה 3

בשאלה זו נעסוק במימוש פעולות אריתמטיות על מספרים שלמים ואי שליליים בייצוג בינארי.
להלן מספר הערות והנחיות התקפות לכלל הסעיפים בשאלה:

- לאורך השאלה נייצג מספרים בינאריים באמצעות מחרוזת המכילה את התווים "0" ו-"1" בלבד.
- לאורך השאלה אין לבצע המרה של אף מספר בינארי לבסיס עשרוני או לכל בסיס אחר. בפרט, אין להשתמש כלל בפונקציות `int` ו-`bin` של פייתון או בפונקציה `convert_base` שראינו בתרגול 3.
- לאורך השאלה, ניתן להניח כי מחרוזת הניתנת כקלט היא "תקינה", כלומר, מכילה אך ורק את התווים "0" ו-"1", וכי התו השמאלי ביותר במחרוזת הוא "1" (מלבד המחרוזת "0" אשר מייצגת את המספר 0). בפרט, מחרוזת למספר שאינו אפס לא תכיל אפסים מובילים והמחרוזת המייצגת את אפס תכיל "0" יחיד.
- לכל פונקציה בשאלה אשר מחזירה כפלט מחרוזת בינארית יש לוודא כי המחרוזת תקינה על פי ההגדרה הקודמת. (למשל, הפלטים "0100" ו-"000" אינם תקינים ואילו הפלטים "100" ו-"0" תקינים).
- לאורך השאלה נעבוד עם מספרים אי-שליליים בלבד. בפרט, ניתן להניח כי המחרוזות הבינאריות הניתנות כקלט לפונקציות השונות מייצגות מספרים אי-שליליים בלבד.
- הרצה של הפונקציות בשאלה על מחרוזות באורך של עד 10 ספרות צריכה להסתיים בזמן קצר (לכל היותר שניה).

סעיף א'

ממשו את הפונקציה $inc(binary)$ (קיצור של increment) אשר מקבלת מחרוזת המייצגת מספר שלם אי שלילי בכתיב בינארי (כלומר מחרוזת המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר מחרוזת המייצגת את המספר הבינארי לאחר תוספת של 1.
להלן המחשה של אלגוריתם החיבור של מספרים בינאריים (בדומה לחיבור מספרים עשרוניים עם נשא (carry)):

```
          1  (carried digits)

        1 0 1 (binary)
+           1
-------------
=       1 1 0
```

<u>הנחיה מחייבת</u>: יש לממש את האלגוריתם בהתאם להמחשה: ישירות באמצעות לולאות.

דוגמאות הרצה:

```
>>> inc("0")
'1'
>>> inc("1")
'10'
>>> inc("101")
'110'
>>> inc("111")
'1000'
>>> inc(inc("111"))
'1001'
```

סעיף ב'

ממשו את הפונקציה $add(bin1, bin2)$ אשר מקבלת שתי מחרוזות המייצגות מספרים אי שליליים שלמים בכתיב בינארי (כלומר מחרוזות המורכבות מאפסים ואחדות בלבד). הפונקציה תחזיר מחרוזת המייצגת את המספר הבינארי המתקבל מחיבור `bin1` ו-`bin2`.
<u>הנחיה מחייבת</u>: יש לממש את האלגוריתם בהתאם להמחשה בסעיף א': ישירות באמצעות לולאה ואין להשתמש בפונקציה `inc`.
דוגמאות הרצה:

```
>>> add("1","0")
'1'
```

```
>>> add("1", "1")
'10'
>>> add("11", "110")
'1001'
```

<u>סעיף ג'</u>

ממשו את הפונקציה $mod\_two(binary, power)$ אשר מקבלת מחרוזת המייצגת מספר בינארי אי שלילי, $binary$, ומספר אי שלילי (מטיפוס `int`), $power$. הפונקציה תחזיר במחרוזת את הייצוג הבינארי של $binary$ מודולו 2 בחזקת $power$, זאת אומרת שהפונקציה תחזיר את הייצוג הבינארי של $binary\ \%\ (2^{power})$.

```
>>> mod_two("11110", 3)
'110'
>>> mod_two("1001", 1)
'1'
>>> mod_two("100", 2)
'0'
>>> mod_two("100", 4)
'100'
```

<u>סעיף ד'</u>

ממשו את הפונקציה $max\_bin(lst)$ אשר מקבלת רשימה lst המכילה k ($k \geq 2$ שלם) מחרוזות המייצגות מספרים שלמים אי שליליים בכתיב בינארי (כלומר מחרוזות המורכבת מאפסים ואחדות בלבד). הפונקציה תחזיר את המחרוזת בעלת הערך המקסימלי (ניתן להניח שכל המחרוזות שונות ותקינות).

```
>>> max_bin(["1010","1011"])
"1011"
>>> max_bin(["10","0","1"])
"10"
```

<u>סעיף ה'</u>

בהרצאה ראינו כי מספר בבסיס b בעל d ספרות דורש בבסיס c מספר ספרות הוא לכל היותר $\lceil d * \log_c b \rceil$.
<u>הוכיחו טענה זו בקובץ ה-PDF.</u>
<u>הערה</u>: אין צורך להסתבך בהוכחה ארוכה, ניתן להוכיח את הטענה בשורות בודדות כאשר מסתמכים על החסמים העליון והתחתון שראינו גם כן בהרצאה לגודלו של מספר בהינתן שהוא בעל n ספרות בבסיס b ועל הנוסחא להחלפת בסיס לוג.

שאלה 4

בשאלה זו נשאף למצוא שעת קבלה שתספק את כולם, אך כמו תמיד – לא בטוח שנצליח.

לצורך הפשטות נתרכז **ביום** ספציפי בשבוע בו אנו מעוניינים לקבוע את השעה.

נייצג **סלוט זמן** (למשל שיעור) ע"י tuple באורך 2 (תחילת השיעור וסוף השיעור, בסדר זה, ובאורך זמן חיובי). לצורך פשטות נניח כי כל שיעור מתחיל ומסתיים בשעה עגולה שתיוצג ע"י מספר שלם ב-tuple. למשל סלוט הזמן שבין 15:00 עד ל-18:00 ייוצג ע"י –

```
(15, 18)
```

נייצג את **מערכת שעות** של סטודנטית ביום המדובר כרשימה `student_schedule` וכל איבר בה מייצג סלוט של זמן בו הסטודנטית בשיעור, לדוגמא:

```
student_schedule = [(8, 10), (10, 12), (15, 18)]
```

במקרה זה הסטודנטית לוקחת שיעור מ-8:00 עד 10:00, לאחר מכן שיעור נוסף מ-10:00 עד 12:00 ומסיימת את היום עם שיעור מ-15:00 עד 18:00. ניתן להניח כי, כמו בדוגמא, אין חפיפות בין השיעורים והרשימה ממוינת לפי זמני תחילת השיעורים.
הרשימה יכולה להיות גם ריקה, במקרה זה אין לסטודנטית שיעורים באותו יום.

כשמתקבלת החלטה על שעת הקבלה, כמובן, יש לשקול מערכות שעות של סטודנטים שונים. נייצג את **אוסף מערכות השעות** שלהם ע"י מילון, `student_schedules_dict`, כך ששם הסטודנט ממופה למערכת השעות שלו. ניתן להניח כי המילון אינו ריק (כלומר מכיל לפחות סטודנט אחד). לדוגמא:

```
student_schedules_dict = {
      'noam': [(8, 10), (10, 12), (15, 18)],
      'larry': [(10, 11), (14, 16)],
}
```

את **שעת הקבלה** ביום המדובר, `office_hour`, נייצג כסלוט-זמן בן שעה אחת.

בשאלה ניתן להניח את תקינות הקלטים על פי הפורמט המתואר.

<u>סעיף א'</u>

ממשו את הפונקציה `assess_office_hour(office_hour, student_schedules_dict)` שמקבלת את זמן שעת הקבלה שברצוננו להעריך, ואת מילון מערכת השעות של הסטודנטים לאותו יום (בפורמט המתואר לעיל). הפונקציה תחזיר tuple אשר: ערכו הראשון הוא רשימת השמות <u>שזמינים</u> להגיע לשעת הקבלה, וערכו השני הוא <u>היחס</u> של הסטודנטים <u>שזמינים</u> להגיע לשעת הקבלה מבין <u>כל</u> הסטודנטים.

<u>דוגמת הרצה:</u>

```
>> office_hour = (11, 12)
>> student_schedules_dict = {
          'noam': [(8, 10), (10, 12), (15, 18)],
          'larry': [(10, 11), (14, 16)],
      }
>> assess_office_hour(office_hour, student_schedules_dict)
(['larry'], 0.5)
```

<u>סעיף ב'</u>

ננסה בכל זאת לספק את כולם. נרצה לדעת באילו שעות במהלך היום נוכל לקיים שעת קבלה עבורה **כל** הסטודנטים זמינים.

<u>סעיף ב'1</u>

ממשו את פונקציית העזר `merge_intervals(intervals)` שמקבלת רשימה של אינטרוולים (interval) שלמים, כלומר כל איבר ברשימה הוא tuple באורך 2 של מספרים שלמים בסדר עולה (דוגמא לאינטרוול: `(-2, 43)`). על הפונקציה להחזיר רשימה של אינטרוולים **זרים** אשר **ממוינת** לפי ערך ההתחלה של כל אינטרוול.

הפונקציה תעשה זאת ע"י מיזוג כל האינטרוולים החותכים זה את זה. למשל, האינטרוול `(-2, 43)` **חותך** את האינטרוול `(20, 52)`, וניתן **למזגם** לאינטרוול `(-2, 52)`.

<u>דוגמת הרצה:</u>

```
>> merge_intervals([(-2, 43), (-700, -9), (20, 52), (52, 60)])
[(-700, -9), (-2, 60)]
>> merge_intervals([(-2, 43), (-700, -9)])
[(-700, -9), (-2, 43)]
>> merge_intervals([(8, 10), (10, 12), (10, 11), (15, 18), (14, 16)])
[(8, 12), (14, 18)]
```

<u>סעיף ב'2</u>

ממשו את הפונקציה `find_perfect_slots(student_schedules_dict)` שמקבלת את מערכות השעות של כל הסטודנטים, ומחזירה את כל הסלוטים בני-שעה שבהם **כל** הסטודנטים זמינים לשעת הקבלה, או רשימה ריקה אם אין כאלה. הניחו כי שעת הקבלה יכולה <u>להתחיל</u> לכל המוקדם ב-7:00 וכן <u>להתחיל</u> לכל המאוחר ב-19:00.

<u>הנחיה:</u> יש להשתמש בפונקציית העזר שמימשתם בסעיף ב'1.

<u>דוגמאות הרצה:</u>

```
>> student_schedules_dict = {
          'noam': [(8, 10), (10, 12), (15, 18)],
          'larry': [(10, 11), (14, 16)],
      }
>> find_perfect_slots(student_schedules_dict)
[(7, 8), (12, 13), (13, 14), (18, 19), (19, 20)]
```

שאלה 5

לפניכם קטע קוד :

```
x = 8
y = 'cs'
z = x
y = x
y += 12
lst1 = [x, z]
lst2 = [x, y, lst1]          # breakpoint 1
def what(x, lst):
      x -= 10
      lst[0] = "i"
      lst2 = [x, y, what]     # breakpoint 2
      return lst2
lst1 = lst1 + lst2           # breakpoint 3
lst3 = what(x, lst1)         # breakpoint 4
```

ציירו והסבירו **בקובץ ה-PDF** את תמונת הזיכרון מבחינת מרחב הכתובות ומרחב השמות לאחר שהמפרש מבצע כל אחת מהשורות בהן מופיעה ההערה `breakpoint`. בשאלה זו ניתן לצייר בכתב יד ולסרוק באופן ברור את הציור. הקפידו שהציור יהיה ברור לחלוטין.

יש לזכור, כפי שנאמר בהרצאה, שהאתר "python tutor" לא תמיד משקף באופן מדויק את תמונת הזיכרון ולכן מומלץ לבחון את הדברים באמצעות בדיקת כתובות בזיכרון. בנוסף, שימו לב שב-`breakpoint 2` יש לצייר את כל תמונת הזיכרון ולא רק את הסקופ (scope) הפנימי של הקריאה לפונקציה.