# תוכנה 1 – אביב 2022/23

## תרגיל מספר 3

**<u>הנחיות כלליות:</u>**

קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

את התרגיל הבא צריך להגיש באופן הבא:

- הגשה במערכת ה-Git תתבצע על פי ההנחיות שראיתם בתרגול 1.
  צרו את ה repository שלכם מתוך הקישור הבא:
  https://classroom.github.com/a/qcntTw-w

  יש לוודא שבתיקיית הגיט שלכם נמצאים הקבצים הבאים:
  a. קובץ פרטים אישיים בשם details.txt המכיל את שם המשתמש שלכם ב Moodle ואת מספר תעודת הזהות שלכם.
  b. קבצי ה- java של התוכניות אותם התבקשתם לממש. בתרגיל הנוכחי ישנם 2 קבצים: ArrayUtils.java ו-StringUtils.java הנמצאים בתיקיית src.

- הגשה במערכת ה Moodle (http://moodle.tau.ac.il/): עליכם להגיש את קובץ הטקסט assignment.txt ובו קישור לgit repository האישי שלכם.

---

**הערות כלליות:**

- הקפידו שחתימות המתודות תהיינה **<u>זהות</u>** לאלו המצוינות בשאלה (אין לשנות את החתימות אשר ניתנו בשלד). כמובן שאת משפטי ה-return שמצויים בשלד אפשר לשנות.
- בתרגיל זה אין צורך לטפל במקרים בהם מערכי\מחרוזות הקלט שווים ל-null אלא אם צוין אחרת.
- **בתרגיל זה עליכם להגיש שתי מחלקות, ולהשלים את הקוד בשלד הנתון. המחלקות לא כוללות מתודת main, ואין להגיש אותן עם מתודת main.**
- כדי לבדוק את עצמכם, כתבו מחלקה נפרדת, בה הוסיפו מתודת main, ובדקו את המחלקות והמתודות בה. **את המחלקה אשר בניתם לצורך בדיקה אין להגיש**.
- כמו כן, לכל תרגיל מצורפות דוגמאות אשר מדגימות מהו הפלט הרצוי. עם זאת, בדיקת התרגיל תעשה על קלטים נוספים\אחרים.
- ניתן להוסיף פקודות import לתחילת הקבצים (**אך מומלץ מאד** שתמחקו לפי ההגשה פקודות import מיותרות – האקליפס מסמן אותן עם קו צהוב ואזהרה).
- כמו כן, נא לא להשתמש אף פעם במתודה System.exit. היא משבשת את הבדיקות האוטומטיות.
- ניתן ומומלץ להוסיף מתודות עזר (מותר ומומלץ גם, אך לא חובה, שהן יהיו private). בתוך אותה מחלקה, ניתן בהחלט להשתמש באותה מתודת עזר לתרגילים שונים. עם זאת, **עליכם להימנע ממצב שבו תתבצע קריאה מקובץ אחד למתודה (גם לא מתודת עזר) מקובץ אחר**. כלומר, כל מתודה שאתם מממשים במטלה זו בקובץ אחד צריכה להיות מסוגלת לרוץ בלי שימוש בקוד שנמצא בקובץ השני. אם אתם רוצים להשתמש באותה מתודת עזר בשני הקבצים, יש לרשום אותה פעם אחת בכל קובץ. נעיר

שבפרוייקט אמיתי היינו נמנעים עקרונית משכפול קוד והיינו מוותרים על הדרישה הנ"ל – דרישה זו נובעת אך ורק עבור פשטות הבדיקה של המטלה.

## חלק א' – מערכים (50 נק')

**<u>בחלק זה מבנה הנתונים היחידי בו מותר להשתמש הינו מערכים.</u>**

ממשו מחלקה בשם **ArrayUtils** שתכיל את המתודות הסטטיות הבאות:

1. **[20 נק']** ממשו מתודה בשם ***shiftArrayCyclic*** המקבלת מערך המכיל מספרים שלמים ומחזירה את המערך המקורי בו איברי מערך הקלט מוזזים לכיוון המצוין כמספר הפעמים שצוין. אם הפרמטר direction שווה ל-'R', המערך יוזז ימינה move פעמים, כך שבכל פעם האיבר האחרון מוזז למקום הראשון במערך (ושאר האיברים זזים מקום אחד ימינה כמובן). ואילו כאשר הפרמטר direction שווה ל-'L', באופן אנלוגי לחלוטין, המערך יוזז שמאלה move פעמים, כך שבכל פעם האיבר הראשון מוזז למקום האחרון במערך. ניתן להניח כי המערך המתקבל אינו null.
אם הכיוון המצוין אינו 'R' או 'L', המתודה תחזיר את המערך ללא שינוי.
אם מספר ההזזות (move) אינו חיובי, המערך יוזז לכיוון הנגדי (רמז: x,R- שקול לx,L ולהפך. האם יש צורך לשכפל קוד?). שימו לב כי מספר ההזזות, move, יכול להיות יותר מגודל המערך (במקרה כזה, שמופיע גם בדוגמאות בהמשך, אפשר לחשב את מספר ההזזות כמודולו אורך המערך, שכן כאשר מספר ההזזות שווה לאורך המערך, אנחנו חוזרים למצב המקורי). **יש להחזיר את המערך <u>המקורי</u>, לאחר שערכי התאים בו השתנו. מותר ליצור מערכי עזר.**
<u>הערה:</u> לא ירדו נקודות על פתרון שאינו אופטימלי מבחינת סיבוכיות. אבל כדאי להשקיע מחשבה ליעילות ובפרט לוודא שסיבוכיות הפתרון אינה אקספוננציאלית

חתימת המתודה:

**public static int**[] shiftArrayCyclic(**int**[] array, int move, char direction)

דוגמא:

shiftArrayCyclic ([1, 2, 3, 4, 5],-1, 'R') -> [2,3,4,5,1]

shiftArrayCyclic ([1, 2, 3, 4, 5], 1, 'R') -> [5,1,2,3,4]

shiftArrayCyclic ([1, 2, 3, 4, 5], 1, 'r') -> [1,2,3,4,5]

shiftArrayCyclic ([1, 2, 3, 4, 5], -2, 'g') -> [1,2,3,4,5]

shiftArrayCyclic ([1, 2, 3, 4, 5], 3, 'L') -> [4,5,1,2,3]

shiftArrayCyclic ([1, 2, 3, 4, 5], -3, 'L') -> [3,4,5,1,2]

shiftArrayCyclic ([0, 8, 9, 5, 6], 6, 'L') -> [8, 9, 5, 6,0]

shiftArrayCyclic ([],3, 'R') -> []

2. **[30 נק']** ממשו מתודה בשם ***findShortestPath*** המקבלת מערך דו מימדי אשר מייצג מטריצת שכנויות של גרף מכוון. כלומר הכניסה ה-[j][i] מכילה את הספרה 1 אם קיימת קשת מכוונת מהקדקוד ה-i לקדקוד ה-j, ו-0 אחרת. זה אומר, במילים אחרות, שיש מסלול באורך 1 מקדקוד i לקדקוד j. שימו לב, כיוון שהגרף מכוון, לא בהכרח מתקיים שהכניסה ה-[j][i] שווה לכניסה ה-[i][j].
בנוסף, שימו לב כי המטריצה הזו היא ריבועית (כלומר מספר התת-מערכים שווה לאורכו של כל מערך, והאורך הזה הוא מספר הצמתים בגרף).
המתודה מקבלת בנוסף שני אינדקסים i, j ומחזירה את **אורך המסלול הקצר** ביותר שמתחיל בקודקוד i (הפרמטר השני שמקבלת המתודה) ומסתיים בקודקוד j (הפרמטר השלישי שמקבלת המתודה). אם לא קיים מסלול בין קודקוד i ל-j, המתודה תחזיר 1-.
אורך מסלול הוא מספר הקשתות בו. ניתן להניח כי המטריצה הינה מטריצה ריבועית אשר עומדת בדרישות, האינדקסים מהווים קלט חוקי וכי המטריצה שונה מ-null.

<u>הערה:</u>

- **מותר להוסיף מתודות עזר במידת הצורך. נסו לחשוב על פתרון יעיל כמה שניתן (ראה הערה על סיבוכיות בסוף ההנחיות לשאלה הקודמת).** מותר, כמובן, אך לא חובה, להשתמש ברקורסיה.
- המסלול הכי קצר מצומת כלשהי אל עצמה הוא באורך 0.
- אמנם אין על כך איסור רשמי, אך מומלץ לא לשנות את המטריצה המקורית במהלך הפתרון, כי זה יכול בקלות להוביל לשגיאות בחישוב.
- יכולים להיות "מעגלים", לדוגמה קשת מ1 ל0 ומ0 ל1.

חתימת המתודה:

```
public static int findShortestPath(int[][] m, int i, int j)
```

דוגמא:

```
findShortestPath ([[0,0,0],[0,0,0],[0,0,0]],0,1) -> -1
findShortestPath ([[0,0,0],[0,0,0],[0,0,0]],1,1) -> 0
findShortestPath ([[0,1,0,0],[0,0,1,0],[0,0,0,1],[1,0,0,0]],0,2) -> 2
findShortestPath ([[0,1,0,1],[0,0,0,1],[0,0,0,0],[0,0,1,0]],0,2) -> 2
```

# חלק ב' – מחרוזות (50 נק')

**<u>בחלק זה המבנה היחידי בו מותר להשתמש הינו מערכים (למען הסר ספק, מחרוזות לא נחשבות בתור מבני נתונים וניתן להשתמש בהן).</u>**

**<u>מותר, אך לא חובה, להשתמש במתודות המובנות במחלקות String ו-Arrays, אם לא נאמר אחרת.</u>**

ממשו מחלקה בשם **StringUtils** שתכיל את המתודות הסטטיות הבאות:

3. **[20 נק']** ממשו מתודה בשם **findSortedSequence** המקבלת מחרוזת קלט הכוללת אותיות אנגליות קטנות ורווחים בלבד, כאשר בין כל שתי מילים יש בדיוק רווח יחיד (ניתן להניח את תקינות הקלט, בין היתר שקלט לא מתחיל עם רווח), ומחזירה את תת-המחרוזת הארוכה ביותר שלה (מבחינת מספר המילים בה) בה המילים ממוינות לפי סדר לקסיקוגרפי עולה (עם רווחים בין המילים). אם יש כמה תת-מחרוזות ממוינות בעלות אותו אורך (שוב – מבחינת מספר המילים), יש להחזיר את זו שקרובה יותר לסוף מחרוזת הקלט. שימו לב שמיון לקסיקוגרפי זה אותו סדר בו המילים מופיעות במילון, ולא מדובר ב"מיון פנימי" כך שהאותיות במילה עצמה ממוינות. בין היתר, נובע שאם מילה A היא רישא של מילה B, אז במיון לקסיקוגרפי A תופיע בהכרח לפני B.
שתי מילים (או יותר) זהות שמופיעות ברצף נחשבות ממוינות לקסיקוגרפית. במקרה בו אין במחרוזת שתי מילים רצופות שממוינות בסדר לקסיקוגרפי עולה (כלומר המחרוזת כולה ממוינת בסדר יורד) יש להחזיר רק את המילה האחרונה (זה לא כלל נוסף, אלא הבהרה למקרה הקצה שנובעת מההנחיה הכללית). אם מחרוזת הקלט היא המחרוזת הריקה, יש להחזיר מחרוזת ריקה גם כן (אך לא null!). מחרוזת שיש בה רק רווחים שקולה למחרוזת ריקה (יש להחזיר מחרוזת ריקה).
מותר, אך כלל לא חובה, להשתמש במתודות מיון "מוכנות" מהספריות הסטנדרטיות של ג'אווה (כמו ב- java.util.Arrays). מותר, אך לא חובה, גם להשתמש במתודת join של String וגם במתודות השוואה של מחרוזות שנלמדו בתרגול.
<u>תזכורת</u>: שוויון בין מחרוזות יש לבדוק עם המתודה equals ואף פעם לא עם ==.
ניתן להניח כי מחרוזת הקלט שונה מ-null. וודאו שאין במחרוזת <u>המוחזרת</u> רווחים לפני המילה הראשונה או אחרי האחרונה, ושיש רווח אחד בדיוק בין כל שתי מילים עוקבות.
   - ✓ רמז: היעזרו בפקודה split של המחלקה String

חתימת המתודה:

**public static String** `findSortedSequence` (**String** str)

דוגמא (שימו לב שלא צריך גרשיים בפלט, והם מופיעים כאן רק כדי לסמן שמדובר במחרוזת):

`findSortedSequence` (“to be or not to be”) -> "not to"

`findSortedSequence` ("my mind is an empty zoo”) -> "an empty zoo"

`findSortedSequence` ("") -> ""

`findSortedSequence` ("andy bought candy") -> "andy bought candy"

`findSortedSequence` ("life is not not not fair") -> "is not not not"

`findSortedSequence` ("art act") -> "act"

4. **[30 נק']** ממשו את המתודה: `isEditDistanceOne` המקבלת שתי מחרוזות ומחזירה true נדרש לכל היותר צעד עריכה אחד על מנת להגיע ממחרוזת אחת לשניה. צעד עריכה יכול להיות אחד מהבאים: הוספת תו, מחיקת תו, החלפת תו.
לדוגמא: עבור המחרוזות cat ו car נדרש צעד עריכה יחיד בשביל לעבור ביניהן (החלפת האות האחרונה). כך גם עבור המחרוזות cat ו ct (ניתן להוסיף אות אחת ל ct, שזה שקול להסרת אות אחת מ cat). עבור המחרוזות cult cat נדרש יותר מצעד עריכה אחד בשביל לעבור מאחת לשניה.
הנחיות למימוש בשאלה זו:
   א. אין להשתמש ברקורסיה. עליכם לממש מימוש איטרטיבי.
   ב. אין לייצר מחרוזות חדשות או מערכים חדשים בזכרון. אין לקרוא למתודות שבמימוש שלהן מתבצעת יצירה של מחרוזת חדשה (כגון המתודה substring במחלקה String) אפילו אם התוצאה של הקריאה למתודה לא נשמרת במצביע (משתנה מטיפוס String) שישמור את המחרוזת. כנ"ל לגבי מערכים.
   ג. מותר לייצר משתנים מקומיים שאינם מחרוזות/מערכים וכן להגדיר מצביעים חדשים למחרוזות הקיימות.

חתימת המתודה:

```
public static boolean isEditDistanceOne(String a, String b)
```

דוגמאות:

```
isEditDistanceOne ("dog","god") -> false

isEditDistanceOne ("x","x") -> true

isEditDistanceOne ("main","man") -> true

isEditDistanceOne ("ab","cab") -> true
```

---

# בהצלחה !