

תוכנה 1 – חורף תשע"ז

תרגיל מספר 3

מערכים ומחרוזות

הנחיות כלליות:

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

- הגשת התרגיל תיעשה במערכת ה-moodle בלבד (<http://moodle.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש aviv יקרא הקובץ aviv_hw3.zip). קובץ ה-zip יכיל:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
 - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש.

הערות כלליות:

- הקפידו שחתימות המתודות תהיינה זהות לאלו המצוינות בשאלה.
- ניתן להוסיף מתודות עזר.
- בתרגיל זה אין צורך לטפל במקרים בהם מערכי־מחרוזות הקלט ריקים או 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.

חלק א' – מערכים (50 נק')

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

בחלק זה מבנה הנתונים היחידי בו מותר להשתמש הינו מערכים.

1. **10 נק'** ממשו מתודה בשם `incArray` המקבלת מערך המכיל מספרים שלמים ומחזירה מערך חדש בו כל איבר i במערך הקלט מוגדל ב i (כאשר i רץ בין 0 ועד גודל המערך פחות 1). ניתן להניח כי המערך המתקבל אינו null, חתימת המתודה:

```
public static int[] incArray (int[] array)
```

דוגמא:

```
incArray ([1, 2, 3, 4, 5]) -> [1,3,5,7,9]
```

```
incArray ([0,0,0,0,0])-> [0,1,2,3,4]
```

2. **10 נק'** ממשו מתודה בשם `sumArray` המקבלת מערך המכיל מספרים שלמים ומחזירה מערך חדש בו איברי מערך הקלט נסכמים בזוגות. האיבר הראשון במערך החדש הוא יהיה הסכום של השניים הראשונים, השני יהיה הסכום של השלישי והרביעי, וכן הלאה. במידה והמערך המקורי באורך איזוגי האיבר האחרון במערך החדש יהיה שווה לאיבר האחרון במערך הישן.

חתימת המתודה:

```
public static int[] sumArray (int[] array)
```

דוגמא:

```
sumArray ([1, 2, 3, 4, 5]) -> [3, 7,5]
```

```
sumArray ([1, 2, 3, 4]) -> [3,7]
```

3. **15 נק'** נגדיר **סכום מתחלף** באופן הבא: בהינתן שני אינדקסים: i, j , הסכום המתחלף הינו האיבר במקום ה- i , פחות האיבר במקום ה- $i+1$, ועוד האיבר במקום ה- $i+2$, וכו'. (עד האינדקס ה- j), כלומר סכום כל האיברים בין שני האינדקסים הנתונים, כאשר כל איבר שני מוחסר מהסכום.

ממשו מתודה בשם `alternateSum` המקבלת מערך המכיל מספרים שלמים, ומחשבת את **הסכום המתחלף המינימלי**. כלומר המתודה בודקת את כל כל הסכומים המתחלפים האפשריים, ומחזירה את מינימלי מביניהם. ניתן להניח כי המערך המתקבל אינו null. שימו לב כי הסכום המתחלף מוגדר להיות אך ורק בין איברים עוקבים.

חתימת המתודה:

```
public static int alternateSum(int[] array)
```

דוגמא:

```
alternateSum([1, -2, 3, 4, 5]) -> -6
```

```
alternateSum([1, 2, -3, 4, 5]) -> -8
```

הערה: מותר להוסיף מתודות עזר במידת הצורך. נסו לחשוב על פיתרון יעיל כמה שניתן

רמז: האם ניתן להשתמש בחישובים שכבר חישבנו?

4. **[15 נק']** ממשו מתודה בשם **matrixMultiplication** המקבלת שני מערכים דו מימדיים המכיל מספרים שלמים, ומייצגים מטריצות מלבניות, ומחשבת את כפל המטריצות. לקריאה כיצד לחשב זאת קראו: [כפל מטריצות](#). המתודה מחזירה את התוצאה. ניתן להניח כי אורכי המטריצות עומדים בדרישות לביצוע הכפל, וכי המטריצות שונות מ-null.

חתימת המתודה:

```
public static int [][] matrixMultiplication (int[][] m, int[][] n)
```

דוגמא:

```
matrixMultiplication ([[1,2,3],[4,5,6],[7,8,9]],[[1,0,0],[0,1,0],[0,0,1]]) -> [[1,2,3],[4,5,6],[7,8,9]]
```

```
matrixMultiplication ([[1,2],[3,4],[5,6]],[[1,1],[2,2]]) -> [[5,5],[11,11],[17,17]]
```

חלק ב' – מחרוזות (50 נק')

בחלק זה מבנה הנתונים היחיד בו מותר להשתמש הינו מערכים.

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

5. **[10 נק']** ממשו מתודה בשם **sortStringWords** המקבלת מחרוזת קלט (הכוללת אותיות אנגליות קטנות ורווחים בלבד), ומחזירה מחרוזת בה מופיעות המילים ממחרוזת הקלט כשהן ממוינות לקסיקוגרפית בסדר עולה (עם רווחים בין המילים) מיון לקסיקוגרפי הוא מיון לפי ערכי האסקי של התווים (מחרוזת תמויין לפני מחרוזת שנייה אם הערך האסקי של התו במקום הראשון שהיא שונה מהמחרוזת השנייה נמוך יותר מהערך האסקי באותו מיקום של המחרוזת השנייה). ניתן להניח כי המחרוזת המתקבלת שונה מ-null. ✓
רמז: היעזרו בפקודה `split` של המחלקה `String`

חתימת המתודה:

```
public static String sortStringWords (String str)
```

דוגמא:

```
sortStringWords("to be or not to be") -> "be be not or to to"
```

6. **[20 נק']** כתוב מתודה בשם: **mergeStrings** המקבלת שתי מחרוזות המורכבות מאותיות אנגליות קטנות בלבד ללא הופעות חוזרות (אות יכולה להופיע בכל מחרוזת פעם אחת בלבד, אבל יכולה להופיע בשניהן). המתודה תחזיר מחרוזת חדשה, המכילה רק את האותיות המופיעות בדיוק באחת מהמחרוזות. אין צורך לבדוק תקינות הקלט. במידה וכל התווים מופיעים בשתי המחרוזות יש להחזיר את המחרוזת הריקה. סדר התווים במחרוזת המוחזרת לפי הסדר שבו הן הופיעו (קודם אלה שהופיעו במחרוזת הראשונה ולאחר מכן מהשנייה).

חתימת המתודה:

```
public static String mergeStrings(String a, String b)
```

דוגמאות:

```
mergeStrings ("boy","girl") -> "boygirl"
```

```
mergeStrings ("catdog","boygirl") -> "catdbyirl"
```

```
mergeStrings ("abcdefg","bcgfhi") -> "adehi"
```

7. [20 נק'] אנגרמה היא שעשוע לשון שבו יוצרים מילה חדשה מערבוב אותיותיה של מילה קיימת, או משפט חדש מערבוב אותיות של משפט קיים. לקריאה נוספת: [אנגרמה](#).

כתבו מתודה בשם *isAnagram* אשר מקבלת שתי מחרוזות, ומחזירה האם אחת המחרוזות מתקבלת על ידי שיכול אותיות של השנייה, כלומר האם המחרוזת השנייה הינה אנגרמה של המחרוזת הראשונה. ניתן להניח כי כל התווים בשתי המחרוזות הן אותיות אנגלית קטנות, או רווחים.

חתימת המתודה:

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

דוגמאות:

```
isAnagram("mother in law","hitler woman") -> true
```

```
isAnagram("software","jeans") -> false
```

בהצלחה !