

תוכנה 1

תרגיל מספר 3

הנחיות כלליות:

- קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
- הגשת התרגיל תעשה במערכת ה VirtualTAU בלבד (<http://virtual2002.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip) קובץ ה zip יכיל:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
 - ב. קבצי ה java של התוכניות אותם התבקשתם לממש.
 - ג. קובץ טקסט עם העתק של כל קבצי ה java
 - ד. קובץ טקסט בשם answers עם התשובות לשאלות

חשוב: בתרגיל זה יש לממש את כל המתודות הנדרשות במחלקה אחת אשר תקרא Assignment03. הנכם מתבקשים להקפיד על חתימה של מתודות לפי המפורט בתרגיל, אחרת ייתכן והשאלה לא תיבדק.

חלק א':

כתבו את המתודה binaryAdder, המקבלת שתי מחרוזות (טיפוסים מסוג String) המייצגות מספרים בייצוג בינארי (http://en.wikipedia.org/wiki/Binary_numeral_system#Addition), ומחזירה מחרוזת של תוצאת החיבור בין מספרים אלה. ניתן להניח כי הקלט תקין (כלומר מחרוזות הקלט מכילות רק את התווים '0' ו-'1'). לא ניתן להניח חסם על אורך מחרוזות הקלט. להלן דוגמאות:

```
binaryAdder("0","0") -> "0"
binaryAdder("0","1") -> "1"
binaryAdder("1","0") -> "1"
binaryAdder("1","1") -> "10"
binaryAdder("10100101","111") -> "10101100"
binaryAdder("1010010100101","1110111") -> "1010100011100"
binaryAdder("11110111001101011001010000000010","110010") ->
"11110111001101011001010000110100"
```

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש.

```
public static String binaryAdder(String a, String b) {
}
}
```

חלק ב':

כתבו את המתודה `areAnagrams`, המקבלת שתי מחרוזות (טיפוסים מסוג `String`). ובודקת האם האחת מתקבלת משיכול אותיות השנייה. ניתן להניח שהקלט מורכב רק מתווים באנגלית, כאשר לצורך שאלה זו, רווחים אינם נחשבים כאות. ניתן להניח כי הקלט תקין, כלומר שהמחרוזות אינן `null`.

להלן מספר דוגמאות:

```
areAnagrams("Debit Card","Bad Credit") -> true
areAnagrams("The eyes","They see") -> true
areAnagrams("Conversation","Voices rant on") -> true
areAnagrams("Radar","Tartar") -> false
```

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש.

```
public static boolean areAnagrams(String a, String b) {
}
```

חלק ג':

נגדיר את קבוצת המראה של מערך בתור קבוצת איברים המופיעים במערך ברצף בסדר כלשהו, ואשר מופיעים גם במיקום אחר במערך אך בסדר הפוך.

נרצה לממש את המתודה `maxMirror` המחזירה את גודל קבוצת המראה המקסימלית במערך.

למשל במערך $\{1, 2, 3, 9, 8, 3, 2, 1\}$ גודל הקבוצה המקסימלית הוא 3 (עבור $\{3, 2, 1\}$).

להלן כמה דוגמאות ריצה של הפונקציה:

```
maxMirror({1, 2, 1, 4}) → 3
maxMirror({7, 1, 2, 9, 7, 2, 1}) → 2
maxMirror({1, 2, 3, 2, 1}) → 5
maxMirror({1, 1, 1}) → 3
maxMirror({1}) → 1
maxMirror({}) → 0
```

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש. בסעיף זה אין התייחסות לסיבוכיות זמן הריצה של האלגוריתם.

```
public static int maxMirror (int [] arr) {}
```

חלק ד':

א. ממשו את השרות `minRun` אשר בהינתן מחרוזת (`string`) מחזיר את הריצה (`run`) הקצרה ביותר במחרוזת. ריצה מוגדרת בתור מספר הפעמים שבו מופיע אותו התו ברצף (השרות מחזיר את אורך הרצף הקצר ביותר במחרוזת).

להלן כמה דוגמאות:

```
minRun("hoopla") -> 1
minRun("aaaabbbbCCCCC") -> 3
minRun("bbbbbbbbbaa") -> 2
minRun("") -> 0
```

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש.

```
public static int minRun(String str) {
}
```

ב. ממשו את השרות `maxRun` אשר בהינתן מחרוזת מחזיר את הריצה הארוכה ביותר במחרוזת.

להלן מספר דוגמאות:

```
maxRun("hoopla") -> 2
maxRun("aaaabbbbCCCCC") -> 6
maxRun("bbbbbbbbbaa") -> 8
maxRun("") -> 0
```

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש.

```
public static int maxRun(String str) {
}
```