

# תוכנה 1

## תרגיל מספר 4

### 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

---

### חלק א':

מטרת חלק זה לתרגל כתיבת שירותים סטטיים לא טריוויאליים בהינתן החוזה של השירות. בכל סעיף מוגדר שירות למימוש עם שני חוזים אפשריים. עליכם לממש את השירות פעמיים, כך שיתאים לחוזים. ברוב המקרים ניתן לפתור את התרגיל בקלות ע"י הוספת פונקציות עזר (אף שהתרגיל לא דורש זאת במפורש).

**הערה:** במידת הצורך, עבור כל זוג פונקציות תוכלו להשתמש באחת הפונקציות לצורך מימוש השנייה.

את כל הפונקציות בחלק זה יש לממש במחלקה אחת שתקרא: Assignment4

1. בהינתן מערך כקלט, החזר מערך המכיל את אותם מספרים, אך מסודר כך שלאחר כל מופע של 4 יש מופע של מספר גדול ממש מ-4. מותר להזיז את כל אברי המערך חוץ מהאברים שערכם 4. להלן מספר דוגמאות:

`fix4({5, 4, 9, 4, 9, 5}) -> {5, 4, 9, 4, 9, 5}`

`fix4({9, 4, 3, 4, 2, 9}) -> {2, 4, 9, 4, 9, 3}`

`fix4({1, 2, 3, 4, 5}) -> {1, 2, 3, 4, 5}`

`fix4({1, 2, 7, 4, 2, 1}) -> {1, 2, 2, 4, 7, 1}`

```

/*
 * @pre occurrences(4, arr) <= occurrences(x, arr) where x > 4
 * @pre arr[arr.length - 1] != 4
 * @post forall 0 <= i < arr.length-1, $prev(arr[i]) == arr[i]
 * @post $ret != arr
 * @post values in $ret are a permutation of values in arr
 * @post forall 0 <= i < $ret.length-2, $ret[i] == 4 ==> $ret[i+1] > 4
 * @post forall 0 <= i < arr.length-1, arr[i] == 4 ==> $ret[i] == 4
 */
public static int[] fix4a(int[] arr) {
}

/*
 * @post ((occurrences(4, arr) <= occurrences(x, arr), where x > 4) AND
 *        (arr[arr.length - 1] != 4))
 *        ==>
 *        ((forall 0 <= i < arr.length-1, $prev(arr[i]) == arr[i]) AND
 *         ($ret != arr) AND
 *         (values in $ret are permutation of values in arr) AND
 *         (forall 0 <= i < $ret.length-2, $ret[i]==4 ==> $ret[i+1]>4) AND
 *         (forall 0 <= i < arr.length-1, arr[i] == 4 ==> $ret[i] == 4))
 */
public static int[] fix4b (int[] arr) {
}

```

2. ממשו פונקציה אשר, בהינתן מערך של מספרים שלמים (integers), מכריעה אם ניתן לבחור תת קבוצה מתוך המערך כך שסכום תת הקבוצה שווה לסכום שאר האיברים במערך? ניתן לפתור בעיה זו ע"י רקורסיה. טיפ: במקום להסתכל על כל המערך, נסתכל על המערך החל מאינדקס start ועד לסופו. הקורא לשירות זה יכול לציין שברצונו לפתור עבור כל המערך ע"י נתינת ערך 0 ל-start. להלן כמה דוגמאות:

```

partition({1,2,3,6}) -> true
partition({1,2,3,7}) -> false
partition({1,2,4,4,3}) -> true
partition({5,6,7,2,3,4,5}) -> true

```

```
partition({5,6,6,2,3,4,5}) -> false
```

```
/*
 * @pre 0 <= start <= nums.length - 1
 * @pre nums != null
 * @pre  $\forall 0 \leq i \leq \text{arr.length} - 1: \text{arr}[i] \in Z$ 
 * @pre  $\exists i_1 \leq i_2 \leq \dots \leq i_k: \sum_{j=1}^k \text{arr}[i_j] = s \wedge i_1 \geq 0 \wedge i_k < \text{start}$ 
 * @post  $(\exists I \subset \{0 \dots \text{arr.length} - 1\}: \sum_{i \in I} \text{arr}[i] = \sum_{j \in \{0 \dots \text{arr.length} - 1\} \setminus I} \text{arr}[j]) \Rightarrow \$ret == true$ 
 * @post
 *  $\neg(\exists I \subset \{0 \dots \text{arr.length} - 1\}: \sum_{i \in I} \text{arr}[i] = \sum_{j \in \{0 \dots \text{arr.length} - 1\} \setminus I} \text{arr}[j]) \Rightarrow \$ret == false$ 
 */
public static boolean partition1(double[] nums, int s, int start) {
```

```
/*
 * @pre 0 <= start <= nums.length - 1
 * @pre  $\exists i_1 \leq i_2 \leq \dots \leq i_k: \sum_{j=1}^k \text{arr}[i_j] = s \wedge i_1 \geq 0 \wedge i_k < \text{start}$ 
 * @post  $(\exists I \subset \{0 \dots \text{arr.length} - 1\}: \sum_{i \in I} \text{arr}[i] = \sum_{j \in \{0 \dots \text{arr.length} - 1\} \setminus I} \text{arr}[j]) \Rightarrow \$ret == true$ 
 * @post
 *  $\neg(\exists I \subset \{0 \dots \text{arr.length} - 1\}: \sum_{i \in I} \text{arr}[i] = \sum_{j \in \{0 \dots \text{arr.length} - 1\} \setminus I} \text{arr}[j]) \Rightarrow \$ret == false$ 
 */
public static boolean partition1(int[] nums, int s, int start) {
```

3. בהינתן מערך של מספרים, החזירו מערך חדש בו כל ערך מהמערך המקורי מופיע בדיוק פעם אחת.  
להלן כמה דוגמאות:

```
unique({1,2,2,3,3,4}) -> {1,2,3,4}
```

```
unique({3,6,8,9,9,21,21}) -> {3,6,8,9,21}
```

```

/*
 * @pre arr != null
 * @pre  $\forall 0 \leq i < arr.length - 1 : \underline{arr}[i] \leq \underline{arr}[i+1]$ 
 * @post  $\$ret.length \leq arr.length$ 
 * @post  $\forall 0 \leq i < arr.length$ 
 *            $\forall 0 \leq j < arr.length, j \neq i$ 
 *            $\underline{\$ret}[i] \neq \underline{\$ret}[j]$ 
 */
public static int[] unique1(int[] arr) {

}

```

```

/*
 * @pre arr != null
 * @post  $\$ret.length \leq arr.length$ 
 * @post  $\forall 0 \leq i < arr.length$ 
 *            $\forall 0 \leq j < arr.length, j \neq i$ 
 *            $\underline{\$ret}[i] \neq \underline{\$ret}[j]$ 
 */
public static int[] unique2(int[] arr) {

}

```

4. ניתנים כקלט מערך של שלמים, encoded, ומחרוזת, text. צריך למצוא האם המערך הוא קידוד חוקי של המחרוזת, עפ"י הכלל הבא: מערך קידוד מכיל קוד תו באינדקס זוגי ומספר באינדקס אי-זוגי. במחרוזת שמקודדת במערך התו הנמצא באינדקס ה- $i$  במערך מופיע  $encoded[i+1]$  פעמים. יש להחזיר true אם המחרוזת המקודדת במערך זהה למחרוזת text. להלן מספר דוגמאות:

```

isEncoded({'a',3,'b',2},{aaabb}) -> true
isEncoded({'a',2,'b',3},{aacc}) -> false
isEncoded({'s',1,'u',1,'p',2},{supp}) -> true

```

```

/*
* @pre  encoded.length % 2 == 0
* @pre  (char)encoded[i] in [a-z,A-Z, ], i is even
* @pre  encoded[i] > 0, i is odd
* @pre  text[i] in [a-z,A-Z, ]
* @post $ret = ( $\forall$  0<=k<text.length  $\exists$  h :
*   text[k] == encoded[h] AND
*   encoded[1] + encoded[3] + ... + encoded[h-1] <= k AND
*   encoded[1] + encoded[3] + ... + encoded[h+1] > k )
*/
public static boolean isEncoded1(int[] encoded, String text) {

}

```

```

/*
* @post (encoded.length % 2 == 0    AND
* (char)encoded[i] in [a-z,A-Z, ], i is even    AND
* encoded[i] > 0, i is odd    AND
* text[i] in [a-z,A-Z, ]) =>
* ($ret = ( $\forall$  0<=k<text.length  $\exists$  h :
*   text[k] == encoded[h] AND
*   encoded[1] + encoded[3] + ... + encoded[h-1] <= k AND
*   encoded[1] + encoded[3] + ... + encoded[h+1] > k ))
*/
public static boolean isEncoded2(int[] encoded, String text) {

}

```

## חלק ב':

חלק זה נועד לתרגל כתיבת חוזים עבור שירותים קיימים. בכל סעיף נתונה פונקציה, עליכם כתוב את החוזה (תנאי קדם ואחר) עבור הפונקציה הנתונה. הניחו כי כל הפונקציות אינן בודקות את הקלט.

1. לוגריתם בבסיס 10

```
public static double log(double d) {  
    ...  
}
```

2. העלאה בריבוע

```
public static double square(double d) {  
    ...  
}
```

3. מחלק משותף גדול ביותר

```
public static int gcd(int a, int b) {  
    ...  
}
```

4. מיון

```
public static void sort(int[] arr) {  
    ...  
}
```

## חלק ג':

בחלק זה נתמודד עם שתי בעיות הקשורות למשחקי מילים.

1. נרצה לבנות אפליקציה אשר תעזור לנו בפתרון תשבצים. לשם כך נבנה מחלקה אשר, בהינתן מילון ודוגמא (pattern) מחזירה את כל המילים במילון המתאימות למחרוזת דוגמא. נממש את

האפליקציה במחלקה שתקרא: DictionaryHelper

הדוגמא היא מחרוזת אשר מכילה אותיות או את התו '-', אשר יסמן כי איננו יודעים איזו אות מופיעה במקום זה.

א. ממשו את הפונקציה findMatches אשר מקבלת מילון (מערך של מילים) ומחרוזת דוגמא ומחזירה מערך המכיל את כל המילים המתאימות לדוגמא שסופקה.

להלן מספר הרצות לדוגמא:

```
findMatches("m-d-m", dictionary) -> { "madam", "modem" }
```

```
findMatches("-n--r-m", dictionary) -> { "anagram", "interim" }
```

```
findMatches("cycl--", dictionary) -> { "cycled", "cycles", "cyclic" }
```

```
public static String[] findMatches(String pattern, String[] dictionary) {  
}
```

ב. השלימו את המימוש כך שניתן יהיה להפעיל את השירות משורת הפקודה (command line). התוכנה הממומשת תקבל שני ארגומנטים: הראשון הוא שם הקובץ המכיל את המילים במילון והשני הוא מחרוזת הדוגמא.

**הערה ראשונה:** ניתן להוריד מילון לדוגמא מאתר הקורס<sup>1</sup>.

**הערה שניה:** ניתן להיעזר במחלקה DictionaryReader, אשר מופיעה באתר הקורס. למחלקה פונקציה יחידה, readFromFile, אשר מקבלת שם קובץ של מילון ומחזירה מערך עם המחרוזות במילון:

```
String[] dictionary = DictionaryReader.readFromFile("dictionary.txt");
```

---

<sup>1</sup> המילון הורד מהאתר: <http://www.mieliestronk.com/wordlist.html>

2. נרצה לבנות אפליקציה למציאת אנאגרמה של משפט כללי. האפליקציה תקבל משפט (כלומר מחרוזת המכילה מספר מילים) ותדפיס סדרה של מילים חוקיות המורכבות מהאותיות שהופיעו במשפט המקורי. במימוש שלנו נסתפק בפלט של אנגרמה אחת עבור משפט קלט אחד. שימו לב: רווחים וסימני פיסוק אינם נחשבים כאות וניתן להתעלם מהם.

עמ"נ לבצע את המטלה ביעילות, נייצג מילה (או משפט) ע"י מערך של מספרים, כאשר במקום ה- $i$  יופיע מספר הפעמים בה האות ה- $i$  בא"ב הלועזי (אנגלי) מופיעה במילה. להלן מספר דוגמאות לייצוג מחרוזות במערך:

abba

2	2	0	0	...	0
---	---	---	---	-----	---

abcccc

1	1	4	0	...	0
---	---	---	---	-----	---

נשים לב, שבהינתן ייצוגים כנ"ל למשפט  $s$  ומילה  $w$ , המילה עשויה להיכלל באנאגרמה של המשפט רק אם ערכי כל תא בייצוג המילה קטנים או שווים מהערכים המתאימים בייצוג המשפט. כלומר, לכל תא  $i$  מתקיים:  $w[i] \leq s[i]$ .

בנוסף, נבחין כי אם המילה  $w$  היא אכן חלק מאנאגרמה של  $s$  אזי נוכל לקבל בעיה "קטנה" יותר בה יש לחפש אנאגרמה עבור משפט אחר, המורכב מאותיות  $s$  למעט אלו שהופיעו ב- $w$ . משפט כזה ייוצג ע"י מערך בו ערך התא במקום ה- $i$  הוא בדיוק  $s[i]-w[i]$ .

על סמך הבחנות אלו נוכל לממש את האלגוריתם הבא עבור משפט  $s$ :

א. נבנה מילון של מילים חוקיות

ב. נייצג את כל מילים במילון עפ"י הייצוג שתואר לעיל

ג. נייצג את  $s$  עפ"י הייצוג הנ"ל

ד. נעבור על רשימת המילים החוקיות ונבדוק האם ניתן לשלבן באנאגרמה:

אם המילה הנוכחית עשויה להיות חלק מאנאגרמה, אזי:

$$(1) \text{ נבנה ייצוג חדש } s', \text{ בו } s'[i]=s[i]-w[i]$$



(2) אם בייצוג  $s$  ערכי כל התאים הם 0, אזי מצאנו אנאגרמה חוקית וסיימנו.

(3) אחרת, נפעיל שוב את שלב ד' (ברקורסיה) עם  $s$ .

עליכם לממש את המחלקה AnagramFinder המיישמת את האלגוריתם המוצע. להלן דוגמאות הרצה של המחלקה:

```
> java AnagramFinder dictionary.txt "most of it"
moot fist
> java AnagramFinder dictionary.txt "challenge me"
helm glen ace
> java AnagramFinder dictionary.txt "dinosaurs"
run adios
```

שאלות ותשובות:

א. מהי רשימת המילים החוקיות?

**תשובה:** נשתמש במילון כמו בסעיף הקודם.

ב. אלו אותיות הן חוקיות?

**תשובה:** כולן, אולם אנו נקודד רק את האותיות המעניינות אותנו. נשתמש באותיות קטנות ונעזר במתודה isLetter של המחלקה Character.

ג. מה לעשות במקרה שיש מספר תשובות חוקיות.

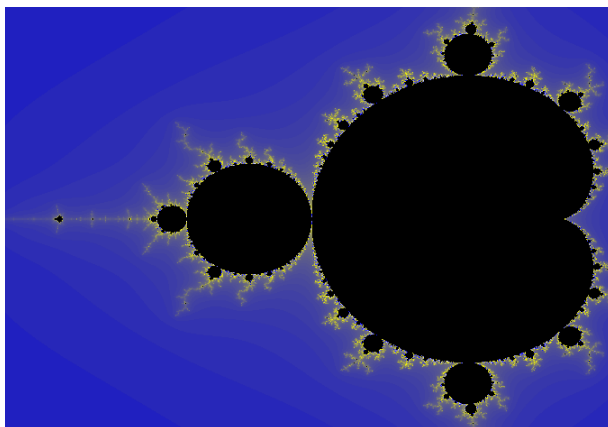
**תשובה:** מספיק לתת פתרון חוקי אחד.

ד. נראה כי ניתן לייעל את הקוד, לדוגמא, מילה ארוכה לא יכולה להיות אנאגרמה של מילה קצרה יותר.

**תשובה:** נכון, ניתן להשתמש באורך המחרוזת כדי לייעל את המימוש. ניתן, לדוגמא להוסיף תא למערך שיכיל את מספר האותיות במילה במיוצגת ולהשתמש בערך זה במהלך החישוב.

## חלק ד':

בחלק זה, נשלים מימוש לאפליקציה לציור הפרקטל המפורסם: Mandelbrot Set.



הפרקטל מתקבל מחישוב חבורת מנדלברוט, הנעשה בצורה הבאה: בהינתן מספר מרוכב,  $c = x + yi$ , נתבונן בסדרה  $\{z\}_{i=0}^{\infty}$  הנתונה על ידי:

$$z_0 = 0$$
$$z_n = z_{n-1}^2 + c$$

ייתכנו שני מקרים:

1. הסדרה מתבדרת (לאינסוף) ובמקרה זה  $c$  אינו חלק מחבורת מנדלברוט.
2. כל איברי הסדרה חסומים ע"י עיגול (כלומר, לכל  $n > 0$  מתקיים:  $|z_n| \leq R^2$ )

מובן כי לא ניתן לחשב את כל איברי הסדרה עבור  $c$  נתון ולכן מסתפקים רק בחישוב מספר מוגבל של איברים מתחילת הסדרה. אם בשלב כלשהוא בחישוב מתקבל מספר החורג מהעיגול המוגדר, עוצרים. לחלופין, אם כל האיברים שחושבו מוכלים בעיגול, מניחים כי  $c$  בסדרה.

**חשוב:** על מנת לענות על שאלה זו, יש להוריד את המחלקה MandelbrotSet מאתר הקורס. המחלקה מספקת ממשק גראפי לציור הפרקטל. כמו כן ניתן לסמן בעזרת העכבר את החלק במישור המרוכב אותו רוצים לצייר. שימו לב כי במחלקה זו הפונקציות המתוארות בסעיפים א' ו-ב' נותרו ללא מימוש. לאחר שתממשו את הפונקציות הללו תוכלו להשתמש באפליקציה.

### סעיף א'

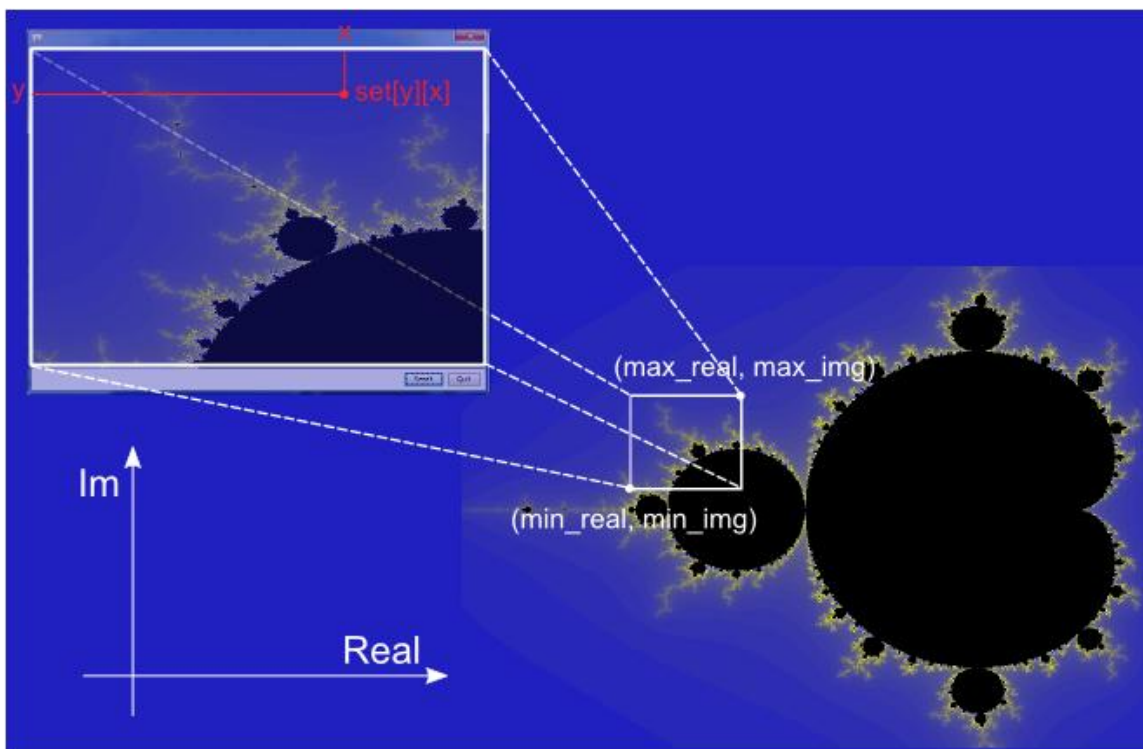
נרצה לממש את הפונקציה calculateMandelbrotAt. הפונקציה מקבלת מספר  $c$  ומחשבת בעבורו את 500 האיברים הראשונים בסדרה, כפי שהוגדרה לעיל. ערך החזרה של הפונקציה יהיה מספר האיברים שחושבו לפני סיום הפונקציה. כלומר, עבור מספרים השייכים לחבורה הפונקציה תחזיר 500 ועבור מספרים שאינם בחבורה הפונקציה תחזיר את  $n$  הקטן ביותר עבורו  $|z_n| > R^2$ . לצרכי התרגיל, נקבע  $R=2$ .

נשים לב: מכיוון שאין ב-Java ייצוג סטנדרטי למספרים מרוכבים, הפונקציה תקבל שני ארגומנטים (מטיפוס double) אשר מייצגים את החלק הממשי והחלק המרוכב של  $c$ , כלומר:  $c = \text{real} + \text{img} \cdot i$

```
public static int calculateMandelbrotAt(double real, double img) {  
}
```

### סעיף ב'

על מנת להציג את הפרקטל בחלון האפליקציה, יש למפות חלק מן המישור המרוכב (המוגדר על ידי המשתמש) לחלון האפליקציה ולצייר את ערכי הפונקציה בכל אחת מהנקודות בחלון זה. ערכים בציר הממשי ימופו לציר ה- $x$  בחלון, ואילו הערכים בציר המדומה ימופו לציר ה- $y$ . בנוסף, מכיוון שהמישור הוא רציף, יש לדגום אותו (בצורה אחידה) כך שיתקבל מספר הנקודות הנדרש למילוי החלון (ראו ציור). גם בחלק זה נייצג מספרים מרוכבים על ידי שני מספרים (מסוג double) האחד מייצג את חלקו הממשי של המספר והשני את חלקו המדומה.



יש להשלים את מימוש המתודה `calculateMandelbrotSet`. תפקיד המתודה הוא למלא את המערך `set`, המייצג את הנקודות בחלון האפליקציה. הערך `set[y][x]` מתאים לנקודה בשורה ה-`y` (מלמעלה) ובעמודה ה-`x`. ניתן להניח כי המערך חוקי וכי כל השורות הן בעלות אורך שווה. החלק במישור המרוכב אותו יש לדגום נתון ע"י שתי נקודות, המייצגות שתיים מפינותיו הקיצוניות (כפי שמודגם בציור).

```
private static void calculateMandelbrotSet(int[][] set,
    double min_real, double min_img,
    double max_real, double max_img) {
}
```

**בהצלחה !**