

בחינה בתוכנה 1

סמסטר א', מועד א', תשע"ג

10/02/2013

דן הלפרין, ניר אטיאס, יעל אמסטרדמר, דביר נתנאלי

הוראות (נא לקרוא!)

- משך הבחינה **שלוש שעות**, חלקו את זמנכם ביעילות.
- אסור השימוש בחומר עזר כלשהו, כולל מחשבוניו או כל מכשיר אחר פרט לעט. בסוף הבחינה צורף לנוחותכם נספח ובו תיעוד מחלקות שימושיות.
- יש לענות על כל השאלות בגוף הבחינה במקום המיועד לכך. המקום המיועד מספיק לתשובות מלאות. יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תיפסל. **תשובות במחברת הבחינה לא תיבדקנה**. במידת הצורך ניתן לכתוב בגב טופס הבחינה.
- יש למלא מספר סידורי (מס' מחברת) ומספר ת.ז. על כל דף של טופס הבחינה.
- ניתן להניח לאורך השאלה שכל החבילות הדרושות יובאו, ואין צורך לכתוב שורות import.
- במקומות בהם תתבקשו לכתוב מתודה (שירות), ניתן לכתוב גם מתודות עזר.
- ניתן להוסיף הנחות לגבי אופן השימוש בשירותים המופיעים בבחינה, ובלבד שאין הן סותרות את תנאי השאלה. יש לתעד הנחות אלו כחוזה (תנאי קדם, תנאי בתר) בתחביר המקובל, שייכתב בתחילת השורות.

לשימוש הבודקים בלבד:

שאלה	א	ב	ג	ד	ה	ו	סה"כ
1							
2							
3							
4							

בהצלחה!

© כל הזכויות שמורות

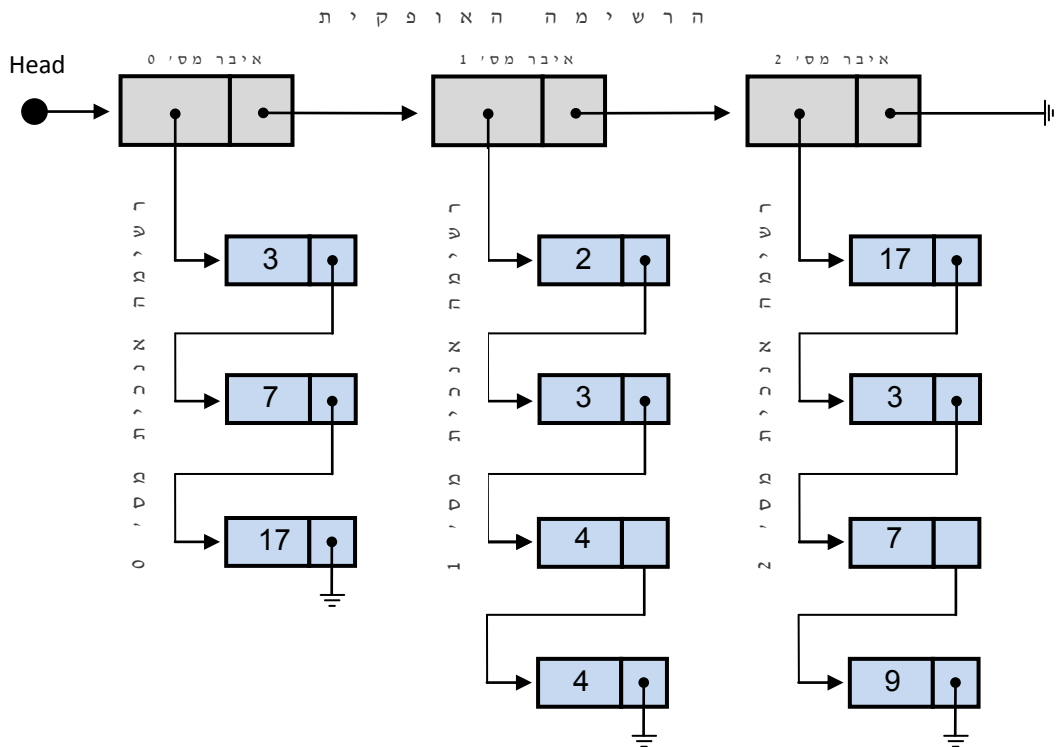
מבלי לפגוע באמור לעיל, אין להעתיק, לצלם, להקליט, לשדר, לאחסן במאגר מידע, בכל דרך שהיא, בין מכאנית ובין אלקטרונית או בכל דרך אחרת כל חלק שהוא מטופס הבחינה.

שאלה 1 (30 נק')

נתון מבנה הנתונים הבא: רשימה מקושרת ראשית, שתיקרא הרשימה האופקית.

מכל איבר ברשימה האופקית משתלשלת רשימה מקושרת – אלה תקראנה הרשימות האנכיות.

ראו הציור.



איבר ברשימה אנכית מכיל נתון מטיפוס גנרי T ומצביע לאיבר מסוג זה:

```
public class Cell<T> {
    private T cont;
    private Cell<T> next;

    public Cell(T cont, Cell<T> next) {
        this.cont = cont;
        this.next = next;
    }

    public T getCont() {
        return cont;
    }

    public Cell<T> getNext() {
        return next;
    }
}
```

בשאלה זו נכתוב כמה מתודות עזר לטיפול ברשימת הרשימות.

א. (15 נקודות) כתבו את המתודה `isWhack`, המקבלת מצביע `head` לראש הרשימה האופקית ושני איברים שונים `t1` ו-`t2` מטיפוס `T`, ומחזירה `int`. המתודה בודקת האם קיימת רשימה אנכית **יחידה** בה מופיעים גם `t1` וגם `t2`. אם כן, המתודה מחזירה את מספרה הסידורי של הרשימה האנכית היחידה בה מופיעים שני האיברים (הרשימה האנכית הראשונה מספרה 0, השניה מספרה 1 וכן הלאה). אחרת, המתודה מחזירה -1.

השלימו את טיפוס המצביע `head` לרשימה האופקית בחתימת המתודה. טיפוס זה יהיה מבוסס גם הוא על המחלקה `Cell` ולא ישתמש באוספים הקיימים בחבילה הסטנדרטית של `Java`. ניתן להניח שלטיפוס `T` יש מתודה `equals` המחזירה `true` כשערך האיברים המשווים שווה (ולא רק כאשר ההפניה זהה).

לדוגמה, עבור רשימה לעיל (בה `T` הוא `Integer`):

`isWhack(head, 3, 4)` תחזיר 1

`isWhack(head, 3, 7)` תחזיר -1

`isWhack(head, 7, 4)` תחזיר -1

`public static <T> int`

`isWhack(` `head, T t1, T t2) {`

- ב. (5 נקודות) נאפיין איבר ברשימה אנכית על ידי זוג אינדקסים:
מספר הרשימה האנכית בה האיבר מופיע מסומן ב- h
מספר האיבר ברשימה האנכית מסומן ב- v
שני האינדקסים מתחילים מ- 0; ראו הציור.
השלימו המתודה equals במחלקה IndexPair להלן

```
public final class IndexPair {
    private int h, v; // horizontal and vertical
                    // indices

    public IndexPair(int h, int v) {
        this.h = h;
        this.v = v;
    }

    @Override
    public boolean equals(Object obj) {

```

```
    }

    @Override
    public int hashCode() {
        int result=17;
        result = result*37+h;
        result = result*37+v;
        return result;
    }
}
```

ג. (10 נקודות) אחסנו את המבנה ב- Map באופן הבא:
כל איבר מטיפוס T במבנה הנתונים הנ"ל ישמר ב- Map כערך (value) המקבל כמפתח (key) את
זוג האינדקסים (h,v).

כתבו מתודה storeInMap המקבלת מצביע head לרשימה אופקית ומחזירה Map כנדרש.

שאלה 2 (40 נקודות)

בשאלה זו נדון בתוכנה לעיבוד תמונה: בחברה לעיבוד תמונה החליטו לפתח חבילת תוכנה שתאפשר, בצורה קלה ונוחה, לשנות תמונה עפ"י הגדרות שונות.

בחבילה זו, תמונה היא בעצם מטריצה דו-מימדית, כאשר הערך במקום ה- x,y הוא בעצם מציין של צבע. ערכי ה- x גדלים משמאל לימין ואילו ערכי ה- y גדלים מלמעלה למטה. כלומר, x הוא קואורדינטת העמודות ואילו y הוא קואורדינטת השורות. לשם פשטות נעסוק רק בתמונות בגווי אפור, בהן הצבע מיוצג ע"י מספר בין 0, המציין את הצבע השחור ל-255 המציין את הצבע הלבן. כל ערך ביניים מייצג רמת אפור מתאימה.



בחבילת התוכנה, תמונות מיוצגות ע"י המנשק:

```
public interface Image {
    public int getPixel(int x, int y);
    public int getWidth();
    public int getHeight();
}
```

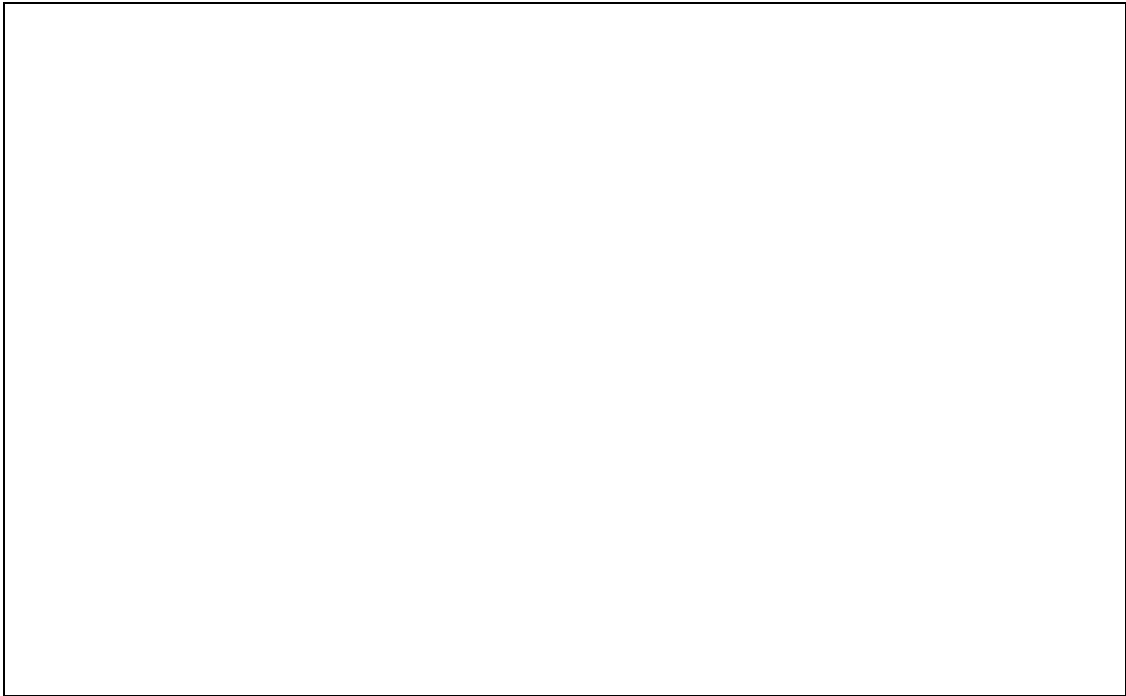
התמרה (טרנספורמציה) של תמונה היא פעולה המשנה את התמונה. בחבילת התוכנה המוצעת, התמרה תמומש במחלקה ייעודית, אשר תממש את המנשק Image באופן הבא: תמונת המקור תועבר להתמרה בבנאי; בכל אחת מן המתודות של ההתמרה נעזר במתודות של תמונת המקור כדי לחשב את הפלט המתאים.

לדוגמא, הקוד הבא משתמש בהתמרה Mirror, המשקפת תמונה לאורך ציר ה- x :

```
Image output = new Mirror(input);
```

תמונה מקורית (input)	תמונה מעובדת (output)
	

א. (8 נקודות) ממשו את המחלקה Mirror הפועלת עפ"י האמור לעיל, כלומר, מקבלת כפרמטר לבנאי תמונה ומממשת את המנשק Image כתמונת שיקוף לאורך ציר x . יש לממש בתוך המלבן המיועד לכך בעמוד הבא.





הוטל על המתכנת דן העצלן לממש את המחלקה Invert המייצגת התמרת תשליל של תמונה נתונה. דן מזהה כי מימוש התמרה זו יהיה דומה מאוד למימוש ההתמרה Mirror ולכן הוא מציע לראש הצוות לכתוב מחלקת התמרות אבסטרקטית אשר תקל על מימוש ההתמרה וכן על מימוש התמרות דומות בעתיד.

ב. (8 נקודות) ממשו את המחלקה AbstractTransform המממשת את שירותי המנשק Image המשותפים להתמרות Mirror ו-Invert.



ג. (4 נקודות) עזרו לדן לממש את התמרת התשליל Invert בעזרת המחלקה האבסטרקטית שהוגדרה בסעיף הקודם. התמרת תשליל מעבירה את ערך הצבע c לערך צבע $255 - c$. בפרט, תחת התמרה זאת, צבע לבן הופך לצבע שחור ולהפך:

תמונה מקורית (input)	תמונה מעובדת (output)
	



ד. המתכנתת טל מזהה כי פעמים רבות איכותה של תמונת הקלט ירודה. היא מציעה לממש את ההתמרה AutoContrast המבצעת תיקון אוטומטי לניגודיות בתמונה, עפ"י האלגוריתם הבא:

1. (5 נקודות) יש לחשב היסטוגרמה של רמות האפור בתמונת הקלט.
 בהיסטוגרמה H, מסכמים את מספר הפיקסלים בתמונה הצבועים בצבע (כלומר בערך אפור) נתון. נייצג היסטוגרמה H ע"י מערך של int כך שהאינדקס ה-i (i בין 0 ל-255) יכיל את מספר הפיקסלים שצבעם i.

```
private static int[] buildHistogram(Image source) {
```

```
}
```


2. יש לחשב את הצבעים הכהה והבהיר ביותר בהיסטוגרמה נתונה. הצבע הכהה ביותר הוא מס' הצבע הנמוך ביותר (הכי קרוב לשחור) שערכו בהיסטוגרמה גדול מ-0. הצבע הבהיר ביותר, בהתאמה, הוא מס' הצבע הגבוה ביותר (הכי קרוב ללבן) שערכו בהיסטוגרמה גדול מ-0.

(5 נקודות) ממשו את הפונקציות `getDarkestColor` ו-`getBrightestColor` המקבלות היסטוגרמה כקלט ומחזירות את הצבע הכהה ביותר והבהיר ביותר בהתאמה. ניתן להניח שבתמונה יש פיקסל אחד לפחות, כלומר, ההיסטוגרמה אינה מכילה רק אפסים.

```
public static int getDarkestColor(int[] histogram) {

}
```

```
public static int getBrightestColor(int[] histogram) {

}
```

3. נסמן ב- p את הצבע הכהה ביותר וב- q את הצבע הבהיר ביותר. יש לחשב את המקדמים a ו- b של הטרנספורמציה הלינארית הממפה את p ל-0 ואת q ל-255. מקדמים אלו נתונים בנוסחאות הללו:

$$a = \frac{255}{q-p}$$

$$b = -255 \cdot \frac{p}{q-p}$$

במידה ו- $p=q$ ניקח את a להיות 1 ואת b להיות 0.

4. בעזרת a ו- b יש להמיר כל פיקסל בעל צבע c בתמונה המקורית לצבע: $a \cdot c + b$.

(10 נקודות) השלימו בעמוד הבא את מימוש ההתמרה `AutoContrast` שהציעה טל. ניתן להשתמש ישירות בפונקציות מהסעיפים הקודמים.

```
public class AutoContrast extends AbstractTransform {
```

```
public AutoContrast(Image image) {
```

```
}
```

```
private void computeAB() {
```

```
}
```

```
public int getPixel(int x, int y) {
```

```
}
```

```
}
```

שאלה 3 (15 נקודות)

קראו את קוד המחלקות הבאות וענו על השאלות.

```
public class A {
    public int a = get();

    public int get() {
        return 0;
    }

    private int get(A[] arr) {
        return arr.length + 1;
    }
}

public class B extends A {
    public int a = 2;

    public int get() {
        return 1;
    }

    public int get(A[] arr) {
        return -1;
    }
}

public class Main {
    public static void main(String[] args) {
        A[] arr = { new A(), new B() };
        /* MISSING CODE */
    }
}
```

בכל אחד מהסעיפים הבאים הוכנס קטע קוד אחר ל-main. הנכם מתבקשים לציין מהו הפלט של הרצת פונקציית ה-main בכל אחד מהמקרים. אם לדעתכם אין פלט לתכנית מכיוון שאינה עוברת קומפילציה או מכיוון שקיימת שגיאה בזמן ריצה (זריקת חריג), הסבירו מה השגיאה. בכל מקרה (שגיאה או ריצה תקינה) יש לכתוב הסבר קצר.

סעיף א' (3 נק')

```
System.out.println(arr[0].a);
```

סעיף ב' (3 נק')

```
int i = arr[1].a;  
System.out.println(arr[i].a);
```

סעיף ג' (3 נק')

```
System.out.println(arr[1].get(arr));
```

סעיף ד' (3 נק')

```
B[] brr = { new B() };  
System.out.println(brr[0].get(brr));
```

סעיף ה' (3 נק')

```
B[] brr = { new A() };  
System.out.println(brr[0].get(arr));
```

שאלה 4 (15 נק')

ברצוננו לכתוב תכנית לספירת המופעים של מילה מסוימת בקובץ טקסט. בשאלה זאת נגדיר מילה חוקית כרצף לא ריק של ספרות ואותיות לועזיות (ללא רווחים או סימנים מיוחדים).

נתונה לנו מתודת העזר `isValid` אשר בודקת אם מחרוזת נתונה היא מילה לא חוקית - כלומר, מחזירה `true` אם מחרוזת הקלט אינה מורכבת מספרות ואותיות בלבד.

```
/**
 * Returns true iff the input contains a character which is not a letter or
 * a digit
 */
private static boolean isValid(String input) {
    ...
}
```

כמו כן, נתונה המתודה `searchForSubWord`. מתודה זו מקבלת כקלט מילה חוקית וחלק טקסט, שניהם ב-`lowercase`, ומחזירה את מס' המופעים של המילה בחלק הטקסט. המילה יכולה להופיע בטקסט כמילה שלמה, או כחלק ממילה אחרת.

```
/**
 * Given a string textPart, and a word, the method returns the amount of
 * times the word appeared in textPart, maybe as a part of another word.
 * @pre word != null
 * @pre isValid(word) == false
 * @pre textPart != null
 */
private static int searchForSubWord(String word, String textPart) {
    int numFound = -1;
    int lastIdx = -1;
    do {
        numFound++;
        lastIdx = textPart.indexOf(word, lastIdx + 1);
    } while (lastIdx != -1);
    return numFound;
}
```

א. (9 נקודות) השלימו את מתודת העזר `readFromFile` אשר מקבלת מסלול לקובץ, קוראת ממנו ומחזירה רשימת מחרוזות המייצגת חלוקה של הטקסט. עליכם להחליט לאילו חלקים לחלק את הטקסט (למשל, לחלק את הטקסט לשורות, או למילים וכו'). את חלקי הטקסט שיצרתם יש לשמור ברשימה ב-`lowercase`. בסעיף הבא נשתמש במתודה `searchForSubWord` כדי לחפש מופעים של מילה נתונה בחלקי הטקסט שתחזיר המתודה.

אין חובה למלא את כל החלקים החסרים, ייתכן יותר מפתרון אחד נכון.

```
/**
 * Gets a file path as an input. Reads the text from the file, splits it by
 * _____ and returns the list of text parts.
 */
private static List<String> readFromFile(String path) throws IOException {
    List<String> text = new LinkedList<>();

```

```
    return text;
}
```

לבסוף, בעמוד הבא מופיעה המתודה הראשית בתכנית, `searchWordInFile`. המתודה מקבלת כקלט מסלול לקובץ טקסט ומילה. תחילה, היא מוודאת שהמילה חוקית. אם כן, היא קוראת את הקובץ הנתון בעזרת `readFromFile`, מחשבת את מספר מופעי המילה בחלקי הטקסט בעזרת `searchForSubWord` ומדפיסה את התוצאה. אם מילת הקלט אינה חוקית או אירעה שגיאה בעת קריאת הקובץ, המתודה מדפיסה הודעה מתאימה וחוזרת.

ב. (6 נק') השלימו את הקוד החסר ב- `searchWordInFile`.

```
public static void searchWordInFile(String path, String word) {
    if (isInvalid(word)) {
        System.out.println("Invalid word " + word
            + ", please try again!");
        return;
    }
    // set the word to lowercase
    word = word.toLowerCase();

    // search for the word in the text of the given file
    List<String> text = null;
```

```
text = readFromFile(path);
```

```
int subWordCount = 0;
for (String textPart : text) {
    subWordCount += searchForSubWord(word, textPart);
}
System.out.println("The word \"" + word
    + "\" appears in the file " + path + " "
    + subWordCount + " times");
}
```