תוכנה 1 – אביב תשע"ד

תרגיל מספר 6

קבצים בינאריים וכתיבת מחלקות

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_hw6.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.

חלק א' (40%) – הצפנת קבצים *עבודה עם קבצים ברמת ה-Byte*

בחלק זה נכתוב את המחלקה EncDec אשר מסוגלת להצפין ולפענח קבצים בשיטה המבוססת על קובץ מפתח. בשיטת ההצפנה זו, אלגוריתם ההצפנה הוא פשוט, אך הוא מסתמך על קיומו של קובץ מפתח המכיל בתים אקראיים (רנדומליים) אותם קשה לשחזר. הן הגורם המצפין והן הגורם המפענח זקוקים לאותו קובץ מפתח על מנת להצליח להצפין ולפענח את הקובץ.

- **שלד המחלקה EncDec נתון לכם באתר הקורס.**
- שימו לב, מעל חלק מהמתודות נתון לכם חוזה המגדיר מה מותר ומה אסור להניח על קלט המתודה. אפשר להניח שהנחות הקדם מתקיימות, ואין צורך לבדוק אותן!
- אין צורך לבדוק תקינות מסלולים לקבצים.
- עליכם לטפל בשגיאות IO בעזרת try ו-catch. אם קרתה שגיאה, יש להשתמש ב-ErrorHelper מתרגיל 4 כדי להודיע על השגיאה למשתמש.
- יש להשתמש ב-ErrorHelper גם כדי לזרוק שגיאה במקרים בהם מוגדר בחוזה שצריכה להיזרק שגיאה.

1. השלימו את מימוש המתודה

```
public void initKey(String filePath)
```

מתודה זו מקבלת מסלול לקובץ מפתח המכיל רצף של בתים (bytes) אקראיים, וקוראת אותו. המפתח יישמר בשדות המופע (שדות לא-סטטיים) של המחלקה EncDec, מכיוון שלכל מופע של EncDec יכול להיות מפתח שונה. כך, נוכל להשתמש בעצם EncDec כדי להצפין ולפענח קבצים רבים עם אותו מפתח.

שימו לב: לפני שאתם מחליטים כיצד לייצג את המפתח, קראו את הסעיפים הבאים וחישבו איזה ייצוג יהיה הנוח והיעיל ביותר במימוש שלכם.

רמז: ניתן להיעזר במתודה `File.length()` כדי לדעת כמה בתים יש בקובץ נתון.

---

2. השלימו את מימוש המתודה

```
public int getKeyLength()
```

מתודה זו מחזירה את אורך המפתח השמור ב-EncDec הנוכחי, או 1- אם לא נטען מפתח עדיין.

---

3. השלימו את מימוש המתודה

```
public void encrypt(String inputFile, String outputFile)
```

מתודה זו מבצעת את תהליך ההצפנה והפענוח, שכפי שמייד נראה, הם זהים לחלוטין.

<u>תיאור תהליך ההצפנה \ הפענוח:</u>

בנוסחה הבאה, נתייחס לקלט, למפתח ולפלט כאל מערכים המכילים מספרים שלמים בין 0 ל-255 -- בדיוק טווח המספרים שניתן לייצג בעזרת בית בודד (8 סיביות).

$$B_{\text{output}}[i] = \left(B_{\text{key}}\left[i \bmod B_{\text{key}}.length\right] - B_{\text{input}}[i]\right) \bmod 256$$

כאשר:

- B_{input} - מערך ערכי הבתים שנקראו מקובץ הקלט.
- B_{output} - מערך ערכי הבתים המוצפנים\מפוענחים שייכתבו לקובץ.
- B_{key} - מערך ערכי הבתים שנקראו מקובץ המפתח.
- $\bmod$ - פעולת מודולו.

שימו לב:

- כל בית בקובץ הקלט ישמש לחישוב בית מתאים בקובץ הפלט. אם קובץ הקלט הוא הקובץ המקורי, תוצאת הפעולה תהיה קובץ מוצפן. אם קובץ הקלט הוצפן ע"י המפתח, תוצאת הפעולה תהיה הקובץ המפוענח.
- במידה והמפתח ארוך יותר מקובץ הקלט אז לא נשתמש בכל הבתים שבו. אולם, אם המפתח קצר יותר, ברגע שנגיע לסופו נתחיל שוב מהתחלה. כלומר, אם המפתח בגודל 200, הבית ה-201 יוצפן בעזרת הבית הראשון של המפתח.
- שימו לב: בג'אווה יש אופרטור % המחשב שארית בחלוקת שלמים. אופרטור זה אינו מתאים לנוסחה שלנו, מכיוון שהפעלתו על מספר שלילי תחזיר מספר שלילי, ואילו אנו מעוניינים במספרים אי-שליליים בלבד. תוצאת החיסור בסוגריים יכולה לצאת שלילית. פתרון אחר שתוכלו לממש הוא הוספה\חיסור של 256 עד שיתקבל מס' בטווח הרצוי (0...256)
- מה יהיה טיפוס המשתנים בנוסחה לעיל ?
  - הפקודה `FileInputStream.read()` קוראת בית בודד מזרם הקלט, ומחזירה אותו כמספר שלם (`int`) בטווח הערכים 0..255.
  - הפקודה `FileInputStream.read(byte[] b)` קוראת הרבה בתים בבת אחת אל מערך מסוג `byte` בו כל מספר הוא בטווח הערכים `-128..127`.

בנוסחה לעיל אנחנו מניחים שערכי הבתים מיוצגים בטווח ערכים של 0-255 ועל כן יהיה קל יותר לבצע את החישוב כולו תוך שימוש במשתנים מסוג `int`. כלומר, ניתן לעבוד עם פקודות IO

הקוראות\כותבות בתים בודדים בתור `int` בשביל פשטות השימוש בנוסחה, למרות מחיר שנשלם מבחינת יעילות הקריאה מקובץ. (שימו לב - ההמרה בין `byte` ל-`int` במקרה הזה **אינה** חיסור או חיבור פשוט, ודורשת פעולות על סיביות!)

דוגמא:
נניח שקובץ המפתח מכיל 3 בתים `[68, 64, 225]` והקובץ אותו אנחנו רוצים להצפין מתחיל ב- "`Once upon a time...`" ולכן מכיל את הבתים המופיעים בשורה האמצעית בטבלה הבאה:

Index:	**0**	**1**	**2**	**3**	**4**	**5**	**6**
B_{key}**`[i mod `**B_{key}**`.length]`**	68	64	225	68	64	225	68
Input (ASCII characters)	O	n	c	e		u	p
B_{input}**`[i]`**	79	110	99	101	32	117	112
B_{output}**`[i]`**	245	210	126	223	32	108	212

אזי לאחר הפעלת הנוסחה המובאת לעיל נקבל את הערכים בשורה התחתונה ביותר בטבלה, ואת הערכים הללו נכתוב לקובץ הפלט. למשל, בעמודה הראשונה `68 - 79 = -11`
ואז, כדי לקבל מס' בטווח הרצוי נוסיף 256 `-11 + 256 = 245`

כעת, אם נפעיל את המתודה על קובץ הפלט עם אותו מפתח, נקבל שוב את קובץ הקלט המקורי!

<u>בדקו את עצמכם</u>

באתר הקורס נתונים לכם קובץ קלט `input.txt`, שני קבצי מפתח `key.dat` ו-`key2.dat`, ו-3 קבצים מוצפנים, `output1.dat`, `output2.dat` ו- `output3.dat`. תוכלו להיעזר בהם כדי לבדוק את התכנית שכתבתם (אין צורך להגיש).

<u>בדיקה I</u>: השתמשו בכל אחד משני המפתחות כדי להצפין את `input.txt`, ושמרו את התוצאה בקובץ חדש. הפעילו שוב את `encrypt` על הקובץ המתקבל -- והתוצאה אמורה להיות זהה ל- `input.txt`

<u>בדיקה II</u>: הצפינו את `input.txt` בעזרת `key.dat` וודאו שהתוצאה זהה ל-`output1.dat`.

<u>בדיקה III</u>: כבר ידוע לנו ש `output1` הוצפן בעזרת `key.dat`, מצאו מי משני המפתחות מפענח את `output2` ו- `output3`... (כאשר מפענחים עם מפתח לא נכון, נקבל קובץ משובש, אקראי ולא קריא)

4. השלימו את מימוש המתודה

```
public void initKey(int size)
```

מתודה זו תייצר מפתח הצפנה אקראי חדש ותשמור אותו ב- EncDec. שדה זה צריך להכיל בדיוק `size` בתים אקראיים. כדי לייצר את המפתח, היעזרו במחלקה `java.util.Random`. שימו לב - מתודה זו היא העמסה של המתודה שכתבתם בסעיף 1, המקבלת ארגומנט מטיפוס שונה.

5. השלימו את מימוש המתודה

```
public void writeKeyToFile(String filePath)
```

מתודה זו תשמור את המפתח הנוכחי לקובץ filePath, כך שניתן יהיה להשתמש בו שוב. כדי לבדוק את עצמכם, השתמשו במתודה initKey מסעיף 1 כדי לטעון מפתח, ואח"כ ב- writeKeyToFile כדי לשמור אותו לקובץ חדש. שני הקבצים אמורים לצאת זהים.

חלק ב' (40%) – קבוצות זרות *מימוש מחלקות לפי מפרט*

בחלק זה של התרגיל נממש מחלקה בשם il.ac.tau.cs.sw1.ex6.DisjointSets המייצגת קבוצה S של קבוצות זרות: $S=\{S_1,S_2,\ldots,S_n\}$ כך שכל S_i הינה קבוצה של מספרים שלמים אי-שליליים, והחיתוך בין הקבוצות ריק (http://en.wikipedia.org/wiki/Disjoint_sets). המחלקה תתמוך בשירותים הציבוריים הבאים:

```
/**
 * Create a singleton set containing x (i.e. {x}) and add it to this object.
 *
 * @pre x>=0
 * @pre x is not in any of the sets Si
 * @post this == $prev(this) U {{x}}
 */
public void makeSet(int x)

/**
 * Return true if and only if x and y belong to the same set in this object.
 *
 * @pre  x in Si && y in Sj
 * @return true iff i == j
 */
public boolean equiv(int x, int y)

/**
 * Find the different sets that x and y belongs to. Remove the sets from
 * this object and add their union
 *
 * @pre  x in Si && y in Sj && i != j
 * @post this = $prev(this) – {Si} – {Sj} U { Si U Sj }
 */
public void joinSets(int x, int y)

/**
 * Return true if and only if x is in a set of this object
 *
 * @pre true ("no precondition")
 * @return true iff exists i such that x is in Si
 */
public boolean inASet(int x)
```

```
 * Return the number of sets in S (each with a different root)
 *
 * @return |S|
 */
public int getNumSets()

/**
 * Return the number of distinct values in S
 *
 * @return |S1|+|S2|+...+|Sn| (n == getNumSets())
 */
public int getNumValues()
```

כמו כן תתמוך המחלקה בבנאי ללא ארגומנטים המייצר קבוצה ריקה של קבוצות זרות.

אפשרות אחת לממש את המחלקה היא ע"י ייצוג כל קבוצה S_i של S בתור עץ.

כל צומת בעץ תייצג מספר שלם אי שלילי x, ותצביע לצומת הורה. בדרך זו, שורש של עץ מייצג בצורה ייחודית את הסט S_i.

מבנה הנתונים הנ"ל ניתן למימוש באמצעות מערך בשם `parent` בו כל תא מייצג צומת בעץ.

האינדקס של כל תא במערך מייצג את המספר המאוחסן בצומת (x), וערך התא (parent[x])משמש כמצביע לאינדקס צומת ההורה. תא במערך שערכו 1- מייצג ערך שלא שייך לאף סט. שורש כל עץ מצביע לעצמו.

אורך המערך צריך להיות גדול יותר מכל מספר x שנמצא כרגע בסט S_i כלשהוא. לפיכך, אם נוסיף מספר גדול יותר מהמקסימאלי עד כה, נצטרך להחליף את המערך במערך חדש גדול יותר. **טיפ:** ניתן להיעזר במתודה System.arraycopy או ב-Arrays.copyOf

כל אובייקט של המחלקה `DisjointSets` מייצג קבוצה של קבוצות זרות, של שלמים אי-שליליים $\{S_1,S_2,...,S_n\}$. ייצוג זה מוגדר פורמאלית ע"י פונקצית העזר getRoot(x):

```
if parent[x] = −1 then getRoot(x) = −1
else if parent[x] == x then getRoot(x) = x
else getRoot(x) = getRoot(parent[x])  //recursive expression
```

כעת, נוכל להגדיר את המיפוי מ-`parent`, שדה של האובייקט, לקבוצה של קבוצות $S = \{S_1,S_2,...,S_n\}$

- איבר שאינו נמצא ב-S כלל:

For all x, [for all i, 1 <= i <= n, x ∉ Si] iff [x >= parent.length or parent[x] == −1]

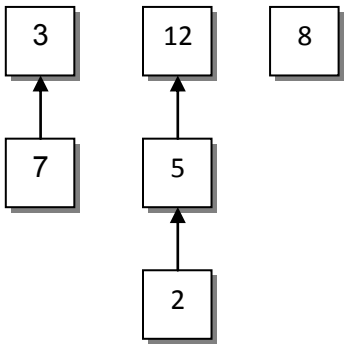
- שני איברים באותו סט S_i:

For all 0 ≤ x,y ≤ parent.length, x,y ∈ Si (x and y are in the same set)
iff getRoot(x) == getRoot(y) and getRoot(x) != −1

למשל, לאובייקט של `DisjointSets` כאשר מערך ה- `parent` הוא:

-1	-1	5	3	-1	12	-1	3	8	-1	-1	-1	12
0	1	2	3	4	5	6	7	8	9	10	11	12

העצים יהיו:

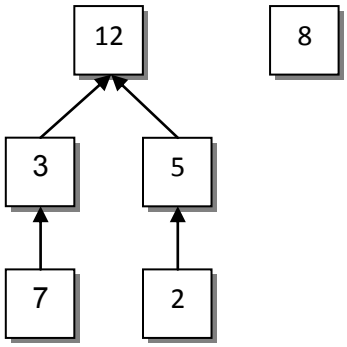


ו- S={ {3,7} {12,5,2} {8}}.

אם נקרא ל-equiv(3,5) נקבל false. אם נפעיל joinSets(7,5) נקבל:

-1	-1	5	12	-1	12	-1	3	8	-1	-1	-1	12
0	1	2	3	4	5	6	7	8	9	10	11	12

העצים יהיו:



(הערה: ייתכן מבנה אחר לעץ השמאלי, אך הוא חייב להיות מורכב מאותם צמתים)

ו-S={{3,7,12,5,2} {8}}.

אם נפעיל כעת makeSet(4) נקבל ש-S={{3,7,12,5,2} {8} {4}}

המשימה:

ממשו את המחלקה `DisjointSets` בצורה יעילה תוך שימוש במערך `parent` (**התבססו על השלד הנתון באתר הקורס**). שימו לב שניתן להגדיר שירותי עזר, ובפרט מומלץ להגדיר:

- פונקציה שמוודאת שמערך ה-`parent` ארוך מספיק להכיל את כל האיברים, ואם לא, מגדילה אותו).
- פונקציה getRoot(int x), אשר תחזיר, עבור כל ערך x, את השורש של העץ של x.

ניתן לבדוק את עצמכם בעזרת פונקציית ה- main הנתונה במחלקה.

חלק ג' (20%) – חידות ג'אווה *תיקון מחלקות קיימות*

בכל סעיף של חלק זה תקבלו חבילה ובה מספר מחלקות. חבילות אלה מצורפות לתרגיל באתר הקורס. עליכם לשנות את הקוד בהתאם להנחיות, כדי לקבל את התוצאה הנדרשת. יש להגיש את כל המחלקות (כולל אלה שלא שיניתם בהם דבר, וכמובן, הקוד המתוקן).

1. החבילה `il.ac.tau.cs.sw1.riddle.a` מכילה שתי מחלקות, `A` ו- `B`.

`B` היא תכנית המקבלת כארגומנט מס' שלם. עליכם <u>לשנות את הקוד</u> בתוך `()printA` ב- `A` בלבד כך שבהרצת פונקציית ה- `main` ב- `B` יודפס המספר שניתן כקלט בין `A1` ל- `A2`. לדוגמא, אם הקלט הוא 15, יודפס:

```
B

A1

15

A2
```

- **מותר:** לשנות את הקוד בתוך `A.printA`.
- **אסור:** לשנות את `B`, את חתימת `()printA`, וקוד ב- `A` שנמצא מחוץ ל- `()printA`.

2. החבילה `il.ac.tau.cs.sw1.riddle.b` מכילה שלוש מחלקות, `A`, `B` ו- `C`.

`C` היא תכנית המקבלת כארגומנטים שלוש מחרוזות. עליכם <u>להשלים את מימוש המתודות</u> `printA`, `printA2` ו- `printA3` ב- `A` כך שבהרצת פונקציית ה- `main` ב- `C` יודפסו 3 המחרוזות בין הכוכביות. לדוגמא, אם הקלט לתכנית הוא `hello world bye`, יודפס:

```
hello

***

world

***

bye
```

- **מותר:** לשנות את הקוד בתוך `printA`, `printA2` ו- `printA3`.
- **אסור:** לשנות את `B` ו- `C`, את חתימות המתודות `printA`, `printA2` ו- `printA3`, וקוד ב- `A` שנמצא מחוץ למתודות הנ"ל.

3. החבילה `il.ac.tau.cs.sw1.riddle.c` מכילה שתי מחלקות, `A` ו- `B`.

`B` היא תכנית. עליכם לשנות את חתימות המתודות והשדות ב- `A` כך ש-(א') הקוד יתקמפל ללא שגיאות, ו-(ב') `B` תדפיס, כשורה אחרונה, `success!`. ייתכן שיודפסו שורות נוספות לפני שורה זו, המשמשות לבקרה בלבד (כל עוד מודפס `success!` הפתרון נכון).

- **מותר:** לשנות את חתימות המתודות ואת השדות, כולל: שינוי נראות (`public` ל-`private` ולהפך), הוספת והורדת `static`, שינוי טיפוס ההחזרה של מתודה, ושינוי הארגומנטים למתודה.
- **אסור:** לשנות את `B`, לשנות קוד בתוך מתודות `A`, לשנות את שמות המתודות ב-`A` ולשנות את ערך השדה `k`.

4. החבילה `il.ac.tau.cs.sw1.riddle.d` מכילה שתי מחלקות, `A` ו- `B`.

`A` היא תכנית המדפיסה מס' שלם. בתוך קוד `A` מופיעות 4 קריאות לפונקציה `setI` של מחלקה `B`, עם הארגומנטים `k`, `B.I`, `j`, ו- `l` בהתאמה. עליכם לשנות את הארגומנטים של `setI` כך שהקוד יתקמפל ללא שגיאות, ובסופו של דבר יודפס המס' 210. עדיין, הארגומנט של כל קריאה חייב להיות `I`, `j`, `k` או `l`.

- **מותר:** לשנות את הארגומנטים של `setI`, למשל במקום בו הופיע `setI(j)` אפשר לשנות ל-`setI(k)`. אם יש צורך ניתן להוסיף שם מחלקה או מופע לפני שדה, למשל `B.I` או `this.l`.
- **אסור:** להשתמש במשתנים\שדות מלבד הארבעה הנ"ל, או במס' טבעיים. אין לשנות את `B` או כל קוד ב-`A` מלבד הארגומנטים המועברים ל- `setI`.

בהצלחה!