מבוא למדעי המחשב – סמסטר א' תשפ"ב

עבודת בית מספר 3: רקורסיה ותכנות מונחה עצמים.

צוות העבודה:

- מרצה אחראי: פרופ' מייק קודיש
- מתרגלים אחראים: רז לפיד ואילן ניימן

תאריך פרסום: 28/11/21

מועד אחרון להגשה: 12/12/21 **בשעה 23:59**.

מה בתגבור:

- בתגבורים של 6-9.12.21 נפתור את משימות: 1.1, 1.2, 4.4

תקציר נושא העבודה:

בעבודת בית זו נכיר את המחלקה BigInteger ואת המחלקה Random, נתרגל רקורסיה ותכנות מונחה עצמים.

בעבודה זו 18 משימות וסך הנקודות המקסימלי הוא 100. הניקוד לכל משימה מפורט במסמך.
העבודה מחולקת לשני חלקים, ובכל אחד מהם משימות בתכנות מונחה עצמים ומשימות רקורסיה.
בעבודה זו **מותר להשתמש בידע שנלמד עד הרצאה 12 (כולל), וכן עד תרגול 7 (כולל).**

הנחיות מקדימות

- קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה. רמת הקושי של המשימות אינה אחידה: הפתרון של חלק מהמשימות קל יותר, ואחרות מצריכות מחשבה וחקירה מתמטית - שאותה תוכלו לבצע בעזרת מקורות דרך רשת האינטרנט. בתשובות שבהן אתם מסתמכים על עובדות מתמטיות שלא הוצגו בשיעורים או כל חומר אחר, יש להוסיף כהערה במקום המתאים בקוד את ציטוט העובדה המתמטית ואת המקור (כגון ספר או אתר).
- עבודה זו תוגש ביחידים במערכת המודל ניתן לצפות בסרטון הדרכה על אופן הגשת העבודה במערכת ה-VPL בלינק הבא: סרטון הדרכה.
- לצורך העבודה מסופקים לכם חמישה קבצים:
  1. Assignment3BigInteger.java
  2. Change.java
  3. NumericalString.java
  4. Bit.java
  5. BitVector.java

המלצה על דרך העבודה- אנו ממליצים לפתוח פרויקט ב-eclipse בשם Assignment3 ולהעתיק אליו את הקבצים.

הנחיות לכתיבת קוד והגשה

- בקבצי השלד המסופקים לכם קיים מימוש ברירת מחדל לכל פונקציה. יש למחוק את מימוש ברירת המחדל בגוף הפונקציות ולכתוב במקום זאת את המימוש שלכם לפי הנדרש בכל משימה.
- אין לשנות את החתימות של הפונקציות המופיעות בקבצי השלד.

- עבודות שלא יעברו קומפילציה במערכת יקבלו את **הציון 0** ללא אפשרות לערער על כך. אחריותכם לוודא שהעבודה שאתם מגישים עוברת תהליך קומפילציה במערכת (ולא רק ב-**eclipse**). להזכירכם, תוכלו לבדוק זאת ע"י לחיצה על כפתור ה-**Evaluate**.
- עבודות הבית נבדקות גם באופן ידני וגם באופן אוטומטי. לכן, יש להקפיד על ההוראות ולבצע אותן במדויק.
- סגנון כתיבת הקוד ייבדק באופן ידני. יש להקפיד על כתיבת קוד יעיל, ברור, על מתן שמות משמעותיים למשתנים, על הזחות (אינדנטציה), ועל הוספת הערות בקוד המסבירות את תפקידם של מקטעי הקוד השונים. אין צורך למלא את הקוד בהערות מיותרות, אך חשוב לכתוב הערות בנקודות קריטיות, המסבירות קטעים חשובים בקוד. הערות יש לרשום אך ורק באנגלית. כתיבת קוד אשר אינה עומדת בדרישות אלו תגרור הפחתה בציון העבודה.

עזרה והנחיה

- לכל עבודת בית בקורס יש צוות שאחראי לה. ניתן לפנות לצוות בשעות הקבלה. פירוט שמות האחראים לעבודה מופיע במסמך זה וכן באתר הקורס, כמו גם פירוט שעות הקבלה.
- ניתן להיעזר בפורום. צוות הקורס עובר על השאלות ונותן מענה במקרה הצורך. שימו לב, **אין לפרסם פתרונות בפורום.**
- בכל בעיה אישית הקשורה בעבודה (מילואים, אשפוז וכו'), אנא פנו אלינו דרך מערכת הפניות, כפי שמוסבר באתר הקורס.
- אנחנו ממליצים בחום להעלות פתרון למערכת המודל לאחר כל סעיף שפתרתם. הבדיקה תתבצע על הגרסה האחרונה שהועלתה (בלבד!).

יושר אקדמי

הימנעו מהעתקות! ההגשה היא ביחידים. אם מוגשות שתי עבודות עם קוד זהה או אפילו דומה - זוהי העתקה, אשר תדווח לאלתר לוועדת משמעת. אם טרם עיינתם בסילבוס הקורס אנא עשו זאת כעת.

משימה 0: הצהרה (0 נקודות)

ב-VPL פתחו את הקובץ readme.txt כתבו בראשו את שמכם ואת מספר תעודת הזהות שלכם.
משמעות פעולה זו היא שאתם מסכימים על הכתוב בו. דוגמה:

I, Israel Israeli (123456789), assert that the work I submitted is entirely my own.
I have not received any part from any other person, nor did I give parts of it for use to others.
I realize that if my work is found to contain code that is not originally my own, a
formal complaint will be opened against me with the BGU disciplinary committee.

חלק 1: שימוש במחלקה BigInteger

מבוא

כחלק מהעבודה עם JAVA בפרט ובעולם פיתוח התוכנה כלל, ישנם כלים רבים שפותחו על ידי מפתחי השפה או קהילת המפתחים.

בחלק זה נכיר שתי מחלקות שהן חלק מ – JAVA:

1. המחלקה BigInteger (מתואר כאן).
2. המחלקה Random (מתואר כאן).

עליכם לקרוא על המחלקות ב - API המפורסם על ידי JAVA, בקישורים למעלה, להכיר את הפונקציות שלהן ולהשתמש בהן. (API - Application Programming Interface)

משימה 1.1 (סכום הערכים הקטנים מ - n) (5 נקודות):

השלימו בקובץ Assignment3BigInteger.java את הגדרת הפונקציה:

```
public static BigInteger sumSmaller(BigInteger n)
```

הפונקציה מקבלת כקלט משתנה n מטיפוס BigInteger ומחזירה משתנה מטיפוס BigInteger שערכו סכום המספרים החיוביים הקטנים מ – n.

הנחייה חובה: עליכם לבצע חישובים (כגון סכימה ובדיקת שוויון) באמצעות הפונקציות של המחלקה BigInteger, כמתואר ב – API.

הנחות על הקלט וחריגות:

- הניחו כי הקלט אינו null.
- פונקציה זו לא זורקת חריגות.

דוגמאות:

```
BigInteger bi0 = new BigInteger("-10");
System.out.println(sumSmaller(bi0)); // 0

BigInteger bi1 = new BigInteger("0");
System.out.println(sumSmaller(bi1)); // 0

BigInteger bi2 = new BigInteger("7");
System.out.println(sumSmaller(bi2)); // 21

BigInteger bi3 = new BigInteger("99999");
System.out.println(sumSmaller(bi3)); // 4999850001
```

משימה 1.2 (עבודה עם המחלקה Random) (5 נקודות):

המחלקה Random מאפשרת לייצר משתנה (מטיפוס Random) אשר באמצעותו יכול המשתמש לבקש מספרים שנדגמים בצורה פסאודו-אקראית. אובייקט מטיפוס Random הוא מחולל (Generator) של מספרים אקראיים.

השלימו בקובץ Assignment3BigInteger.java את הגדרת הפונקציה:

```
public static void printRandoms(int n)
```

הפונקציה מקבלת כקלט משתנה n מטיפוס int ומדפיסה למסך n מספרים מטיפוס int שנדגמו בצורה פסאודו-אקראית ואחידה מכל טווח הערכים האפשריים של ערך מטיפוס int. על כל ערך להיות מודפס בשורה נפרדת.

הנחייה חובה: קראו על הפונקציה ()nextInt ב - API של Random. עליכם לייצר משתנה מטיפוס Random לביצוע המשימה.

הנחות על הקלט וחריגות:
- הניחו כי n אינו שלילי, כלומר $n \geq 0$.
- פונקציה זו לא זורקת חריגות.

דוגמא, פלט אפשרי של הקריאה:

```
printRandoms(5);

// -269001551
// 1230791088
// 1983672709
// -205636269
// 222626083
```

משימה 1.3 (בדיקת ראשוניות של ערך מטיפוס BigInteger) (5 נקודות):

השלימו בקובץ Assignment3BigInteger.java את הגדרת הפונקציה:

```
public static boolean isPrime(BigInteger n)
```

הפונקציה מקבלת כקלט משתנה n מטיפוס BigInteger שערכו אינו שלילי ומחזירה ערך בוליאני המשקף האם הקלט הוא ראשוני או לא, באופן ודאי לחלוטין, כלומר ללא שום סיכוי לשגיאה (שימו לב כי 0 ו-1 אינם ראשוניים).
עליכם לפתור שאלה זו על בסיס האלגוריתם שהוצג בכיתה לבדיקת ראשוניות.

הנחייה חובה: עליכם לבצע חישובים (כגון מודולו) באמצעות הפונקציות של המחלקה BigInteger, כמתואר ב – API.

הנחות על הקלט וחריגות:
- הניחו כי n אינו null וערכו אינו שלילי.
- פונקציה זו לא זורקת חריגות.

<u>משימה 1.4</u> <u>(הגרלת ראשוני קטן מ - 2^n) (5 נקודות):</u>

השלימו בקובץ Assignment3BigInteger.java את הגדרת הפונקציה:

```
public static BigInteger randomPrime(int n)
```

הפונקציה מקבלת כקלט משתנה n מטיפוס int, שערכו גדול ממש מ-1, ומחזירה מספר ראשוני, **שנבחר באופן אקראי**, הקטן מ - 2^n. שימו לב – עליכם להחזיר מספר שהוא באופן ודאי לחלוטין ראשוני (בפרט המספר גדול ממש מ – 1).

הנחייה: אתם רשאים להשתמש בפונקציה מהסעיף הקודם. שימו לב לבנאי של המחלקה BigInteger שחתימתו BigInteger(int numBits, Random rnd).

הנחות על הקלט וחריגות:

- הניחו כי $n > 1$.
- פונקציה זו לא זורקת חריגות.

חלק 2: רקורסיה – בעיית העודף

שימו לב: אין קשר בין הסעיפים בחלק זה.

מבוא

בשאלה זו נעסוק בבעיית העודף. בהינתן רשימת ערכי מטבעות וערך שאותו רוצים לפרוט, הבעיה עוסקת בפריטת הערך הנתון לפי ערכי המטבעות האפשריים.

משימה 2.1 (בעיית העודף) (5 נקודות):

השלימו בקובץ Change.java את הגדרת הפונקציה:

```
public static boolean change(int[] coins, int n)
```

הפונקציה מקבלת כקלט מערך ממוין coins מטיפוס int[] של מספרים חיוביים שונים זה מזה, וערך שאינו שלילי n. הפונקציה מחזירה ערך true אם ניתן לפרוט את n באמצעות מטבעות שערכיהם נמצאים במערך coins, כאשר מותר להשתמש בכמות בלתי מוגבלת של מטבעות מהמערך coins, אחרת הפונקציה מחזירה ערך false.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הנחות על הקלט וחריגות:

- הניחו שהמערך הנתון אינו null, הוא יכול להכיל אך ורק מספרים חיוביים שונים זה מזה והוא ממוין.
- הניחו ש - $n \geq 0$.
- פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון **שאינו** רקורסיבי **לא** יקבל ניקוד.

דוגמאות:

```
int[] coins = {1, 5, 10};

int n = 7;

System.out.println(change(coins, n)); // true
```

הפונקציה תחזיר את הערך true, כיוון שניתן לפרוט את 7 עם סוגי המטבעות הנתונים.

```
int[] coins = {2, 10, 20, 100};

int n = 15;

System.out.println(change(coins, n)); // false
```

הפונקציה תחזיר את הערך false, כיוון שלא ניתן לפרוט את 15 באמצעות מטבעות שערכיהם נמצאים במערך coins.

משימה 2.2 (בעיית העודף עם הגבלת מטבעות – האם קיים פתרון?) (5 נקודות):

השלימו בקובץ Change.java את הגדרת הפונקציה:

```
public static boolean
changeLimited(int[] coins, int n, int numOfCoinsToUse)
```

הפונקציה מקבלת כקלט מערך ממוין coins מטיפוס int[] של מספרים חיוביים שונים זה מזה, ערך שאינו שלילי n וערך שאינו שלילי numOfCoinsToUse.
הפונקציה מחזירה ערך true אם ניתן לפרוט את n באמצעות מטבעות שערכיהם נמצאים במערך coins באמצעות **לכל היותר** numOfCoinsToUse מטבעות שערכיהם נמצאים במערך *coins*, אחרת הפונקציה מחזירה ערך false.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הנחות על הקלט וחריגות:

- הניחו שהקלט תקין, כלומר שהמערך הנתון אינון null, הוא יכול להכיל אך ורק מספרים חיוביים שונים זה מזה והוא ממוין.
- הניחו כי $n \geq 0$ וגם $numOfCoinsToUse \geq 0$.
- פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון **שאינו** רקורסיבי **לא** יקבל ניקוד.

דוגמאות:

```
int[] coins = {1, 7, 12, 19};

int n = 20;

int numOfCoinsToUse = 2;

System.out.println(changeLimited(coins,      n,      numOfCoinsToUse));
// true
```

הפונקציה תחזיר את הערך true, כיוון שניתן לפרוט את 20 עם שני מטבעות בלבד המופיעים במערך.

```
int[] coins = {5, 7, 12};

int n = 8;

int numOfCoinsToUse = 2;

System.out.println(changeLimited(coins,      n,      numOfCoinsToUse));
// false
```

הפונקציה תחזיר את הערך false, כיוון שלא ניתן להגיע לסכום 8 בעזרת המטבעות המופיעים במערך.

```
int[] coins = {1, 7, 10, 12};

int n = 10;

int numOfCoinsToUse = 5;
```

```
System.out.println(changeLimited(coins,      n,      numOfCoinsToUse));
// true
```

הפונקציה תחזיר את הערך true, כיוון שקיים צירוף של מטבע אחד (לכל היותר 5) כך שסכומו יהיה 10. (שימו לב כי קיים צירוף נוסף, 7+1+1+1 ובו 4 מטבעות שסכומם 10).

<u>משימה 2.3</u> <u>(בעיית העודף עם הגבלת מטבעות – הדפסת פיתרון אחד) (5 נקודות):</u>

השלימו בקובץ Change.java את הגדרת הפונקציה:

```
public static void
printChangeLimited(int[] coins, int n, int numOfCoinsToUse)
```

הפונקציה מקבלת כקלט מערך ממוין coins מטיפוס int[] של מספרים חיוביים שונים זה מזה, ערך שאינו שלילי n וערך שאינו שלילי numOfCoinsToUse.
הפונקציה מדפיסה למסך **פתרון אחד** (כלשהו) לפריטה ממוינת לפי ערכי המטבעות, אם ניתן לפרוט את n באמצעות **לכל היותר** numOfCoinsToUse מטבעות שערכיהם נמצאים במערך *coins*, אחרת הפונקציה אינה מדפיסה דבר.
עליכם להדפיס את הערכים מופרדים על ידי פסיק, ללא רווחים, ראו דוגמא.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הנחות על הקלט וחריגות:
- הניחו שהקלט תקין, כלומר שהמערך הנתון אינון null, הוא יכול להכיל אך ורק מספרים חיוביים שונים זה מזה והוא ממוין.
- הניחו כי $n \geq 0$ וגם $numOfCoinsToUse \geq 0$.
- פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון **שאינו** רקורסיבי **לא** יקבל ניקוד.

דוגמאות:

```
int[] coins = {1, 2, 3};

int n = 4;

int numOfCoinsToUse = 2;

printChangeLimited(coins, n, numOfCoinsToUse);
```

הפונקציה תדפיס למסך: 2,2 **או** 1,3

- שימו לב שהמחרוזת המודפסת ממוינת לפי ערכי המטבעות.
- שימו לב שאין פסיק בסוף המחרוזת
- שימו לב שאין רווחים

```
int[] coins = {1, 7, 12};

int n = 10;

int numOfCoinsToUse = 2;
```

```
printChangeLimited(coins, n, numOfCoinsToUse);
```

הפונקציה לא תדפיס דבר.

<u>משימה 2.4</u> <u>(בעיית העודף עם הגבלת מטבעות – מציאת מספר האפשרויות) (5 נקודות):</u>

השלימו בקובץ Change.java את הגדרת הפונקציה:

```
public static int
countChangeLimited(int[] coins, int n, int numOfCoinsToUse)
```

הפונקציה מקבלת כקלט מערך ממוין coins מטיפוס int[] של מספרים חיוביים שונים זה מזה, ערך שאינו שלילי n וערך שאינו שלילי numOfCoinsToUse.
הפונקציה מחזירה את מספר האפשרויות השונות לפרוט את n באמצעות **לכל היותר** numOfCoinsToUse מטבעות שערכיהם נמצאים במערך *coins*. אם לא ניתן לעשות זאת, יוחזר הערך 0.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הנחות על הקלט וחריגות:

- הניחו שהקלט תקין, כלומר שהמערך הנתון אינון null, הוא יכול להכיל אך ורק מספרים חיוביים שונים זה מזה והוא ממוין.
- הניחו כי $n \geq 0$ וגם $numOfCoinsToUse \geq 0$.
- פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון **שאינו** רקורסיבי **לא** יקבל ניקוד.

דוגמאות:

```
int[] coins = {1, 2, 3};

int n = 4;

int numOfCoinsToUse = 2;

System.out.println(countChangeLimited(coins, n, numOfCoinsToUse)); // 2
```

הפונקציה תחזיר את הערך 2, כיוון שישנן שתי דרכים לפרוט 4 באמצעות לכל היותר 2 מטבעות שערכיהם במערך הנתון:
דרך ראשונה: 2,2
דרך שנייה: 1,3

- שימו לב שהאפשרות להחזיר 3,1 שקולה לאפשרות להחזיר 1,3 ולכן זו אינה נספרת כדרך נוספת.

```
int[] coins = {5, 10, 20, 50, 100};

int n = 100;

int numOfCoinsToUse = 5;

System.out.println(countChangeLimited(coins, n, numOfCoinsToUse)); // 6
```

הפונקציה תחזיר את הערך 6 כיוון שישנן שש דרכים לפרוט 100 באמצעות לכל היותר 5 מטבעות שערכיהם במערך הנתון:
דרך ראשונה: 20,20,20,20,20
דרך שנייה: 10,10,10,20,50
דרך שלישית: 5,5,20,20,50
דרך רביעית:100
דרך חמישית: 50,50
דרך שישית: 10,20,20,50

```
int[] coins = {5, 10, 50};

int n = 65;

int numOfCoinsToUse = 2;

System.out.println(countChangeLimited(coins, n, numOfCoinsToUse)); // 0
```

הפונקציה תחזיר את הערך 0, כיוון שלא קיימות דרכים לפרוט 65 באמצעות לכל היותר 2 מטבעות שערכיהם במערך הנתון.

<u>משימה 2.5</u> <u>(בעיית העודף עם הגבלת מטבעות – הדפסת כל האפשרויות) (5 נקודות):</u>

השלימו בקובץ Change.java את הגדרת הפונקציה:

```
public static void
printAllChangeLimited(int[] coins, int n, int numOfCoinsToUse)
```

הפונקציה מקבלת כקלט מערך ממוין coins מטיפוס int[] של מספרים חיוביים שונים זה מזה, ערך שאינו שלילי n וערך שאינו שלילי numOfCoinsToUse.
הפונקציה מדפיסה למסך את כל האפשרויות השונות לפרוט את n באמצעות **לכל היותר** numOfCoinsToUse מטבעות שערכיהם נמצאים במערך *coins*.
כל אפשרות לפריטה מודפסת **ממוינת** לפי ערכי המטבעות, בשורה נפרדת. ערכי המטבעות מופרדים על ידי פסיק, ללא רווחים, ראו דוגמא.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הנחות על הקלט וחריגות:
- הניחו שהקלט תקין, כלומר שהמערך הנתון אינון null, הוא יכול להכיל אך ורק מספרים חיוביים שונים זה מזה והוא ממוין.
- הניחו כי $n \geq 0$ וגם $numOfCoinsToUse \geq 0$.
- פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון **שאינו** רקורסיבי **לא** יקבל ניקוד.

דוגמאות:

```
int[] coins = {1, 2, 3};

int n = 4;
```

```
int numOfCoinsToUse = 2;

printAllChangeLimited(coins, n, numOfCoinsToUse);
```

הפונקציה תדפיס למסך את שתי השורות הבאות בסדר כלשהו:

2,2

1,3

- שימו לב שהמחרוזות ממוינת לפי ערכי המטבעות.
- שימו לב שאין פסיק בסוף המחרוזת
- שימו לב שאין רווחים

```
int[] coins = {1, 5, 10, 20};

int n = 13;

int numOfCoinsToUse = 2;

printAllChangeLimited(coins, n, numOfCoinsToUse);
```

הפונקציה לא תדפיס דבר.

חלק 3: מספרים גדולים בייצוג של מחרוזות

מבוא

חלק זה של העבודה עוסקת בייצוג מספרים (גדולים) בעזרת מחרוזות.

הגדרה: *מחרוזת מספרית* בבסיס $b > 1$ היא מחרוזת שמייצגת מספר בבסיס b. מחרוזת s מייצגת מספר **אי שלילי** בבסיס b אם מתקיימים התנאים הבאים:

א. המחרוזת s *אינה* null ואינה ריקה.
ב. כל התווים במחרוזת הן ספרות חוקיות בבסיס b.
ג. התו האחרון במחרוזת אינו הספרה '0', אלא רק במקרה שיש ב – s רק ספרה אחת.

הייצוג שלנו למספרים בתצורת מחרוזות היא לפי גישה שנקראת least significant bit first (lbsf). בייצוג זה הספרה הפחות משמעותית (זו שמייצגת את ה"אחדות") היא התו הראשון במחרוזת. כלומר, התו הראשון במחרוזת מייצג כפולה של b^0, התו השני במחרוזת מייצג כפולה של b^1, התו השלישי מייצג כפולה של b^2, וכך הלאה. לדוגמא המחרוזת "0111" שהיא מחרוזת מספרית בבסיס 2, מייצגת את המספר 14 שכן
$14 = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3$. כמוכן, המחרוזת "129" היא מחרוזת מספרית בבסיס 10 מייצגת את המספר 921.

הנחיה: בתשובות לחלק זה של העבודה אין להשתמש בשיטות המחלקה BigInteger או כל מחלקה אחרת שלא אתם יצרתם ואשר מיועדת לייצוג מספרים גדולים.

בכל סעיפי חלק זה מותר להשתמש בפונקציה הבאה:

```
public static int toInt(char c) {
     return "0123456789".indexOf(c);
}
```

משימה 3.1 (האם מחרוזת מספרית היא בבסיס b) (4 נקודות):

השלימו בקובץ NumericalString.java את הגדרת הפונקציה:

```
public static boolean legalNumericString(String s, int b)
```

הפונקציה מקבלת כקלט מחרוזת s ומספר b כך ש - $2 \le b \le 10$.
קלט לפונקציה זו נחשב תקין אם s הינה **מחרוזת מספרית** וגם $2 \le b \le 10$. אם הקלט תקין, הפונקציה מחזירה ערך true אם s היא מחרוזת מספרית בבסיס b וערך false אחרת. **אם הקלט לבסיס b הינו $b > 10$ או $b < 2$ נזרוק חריגה מטיפוס IllegalArgumentException עם הודעה מתאימה לבחירתכם.**

דוגמאות:

```
String s = "72849";

int b = 2;

System.out.println(legalNumericString(s, b)); // false
```

הפונקציה תחזיר ערך false מאחר וישנן ספרות שלא בבסיס 2.

<u>משימה 3.2</u> <u>(הוספת 1 למחרוזת מספרית בבסיס 10) (12 נקודות):</u>

השלימו בקובץ NumericalString.java את הגדרת הפונקציה:

```
public static String decimalIncrement(String s)
```

הפונקציה מקבלת כקלט מחרוזת s שהיא מחרוזת מספרית בבסיס 10.
הפונקציה מחזירה מחרוזת מספרית בבסיס 10 שמייצגת את הערך של s ועוד אחד. אם s אינה מחרוזת מספרית בבסיס 10 אזי יש לזרוק חריגה מטיפוס IllegalArgumentException עם הודעה מתאימה לבחירתכם.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

דוגמאות:

```
String s = "5";

System.out.println(decimalIncrement(s)); // "6"
```

הפונקציה תחזיר את המחרוזת – 6.

```
String s = "4321";

System.out.println(decimalIncrement(s)); // "5321"
```

הפונקציה תחזיר את המחרוזת – 5321.

```
String s = "";

for (int i=0; i<70; i=i+1) {

      s = s+"9";

}

System.out.println(decimalIncrement(s));

// "0000000000000000000000000000000000000000000000000000000000000000000000001"
```

הפונקציה תחזיר את המחרוזת –
.0000000000000000000000000000000000000000000000000000000000000000000000001

<u>משימה 3.3</u> <u>(הכפלה ב – 2 למחרוזת מספרית בבסיס 10) (12 נקודות):</u>

השלימו בקובץ NumericalString.java את הגדרת הפונקציה:

```
public static String decimalDouble(String s)
```

הפונקציה מקבלת כקלט מחרוזת s שהיא מחרוזת מספרית בבסיס 10.
הפונקציה מחזירה מחרוזת מספרית בבסיס 10 שמייצגת את הערך הכפול לזה של s. אם s אינה מחרוזת מספרית בבסיס 10 אזי יש לזרוק חריגה מטיפוס IllegalArgumentException עם הודעה מתאימה לבחירתכם.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

דוגמאות:

```
String s = "5";

System.out.println(decimalDouble(s)); // "01"
```

הפונקציה תחזיר את המחרוזת – 01.

```
String s = "4321";

System.out.println(decimalDouble(s)); // "8642"
```

הפונקציה תחזיר את המחרוזת – 8642.

```
String s = "";

for (int i=0; i<70; i=i+1) {

     s = s+"9";

}

System.out.println(decimalDouble(s));

// "899999999999999999999999999999999999999999999999999999999999999999999991"
```

הפונקציה תחזיר את המחרוזת –
899999999999999999999999999999999999999999999999999999999999999999999991.

<u>משימה 3.4</u> <u>(המרת מחרוזת מספרית בבסיס 2 למחרוזת מספרית בבסיס 10) (12 נקודות):</u>

השלימו בקובץ NumericalString.java את הגדרת הפונקציה:

```
public static String binary2Decimal(String s)
```

הפונקציה מקבלת כקלט מחרוזת s שהיא מחרוזת מספרית בבסיס 2.
הפונקציה מחזירה מחרוזת מספרית בבסיס 10 שמייצגת את אותו הערך שמייצגת s. אם s אינה מחרוזת מספרית בבסיס 2 אזי יש לזרוק חריגה מטיפוס IllegalArgumentException עם הודעה מתאימה לבחירתכם.

הנחייה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

דוגמאות:

```
String s = "0";

System.out.println(binary2Decimal(s)); // "0"
```

הפונקציה תחזיר את המחרוזת – 0.

```
String s = "1";

System.out.println(binary2Decimal(s)); // "1"
```

הפונקציה תחזיר את המחרוזת – 1.

```
String s = "11111111";

System.out.println(binary2Decimal(s)); // "552"
```

הפונקציה תחזיר את המחרוזת – 552.

```
String s = "011111111";

System.out.println(binary2Decimal(s)); // "015"
```

הפונקציה תחזיר את המחרוזת – 015.

```
String s = "1011";

System.out.println(binary2Decimal(s)); // "31"
```

הפונקציה תחזיר את המחרוזת – 31.

```
String s = "0011";

System.out.println(binary2Decimal(s)); // "21"
```

הפונקציה תחזיר את המחרוזת – 21.

```
for (int i=0; i<100; i=i+1) {

     s = s+"1";

}

System.out.println(binary2Decimal(s));
// "5735023076941049228220060567621"
```

הפונקציה תחזיר את המחרוזת – 5735023076941049228220060567621.

חלק 4: המחלקות Bit ו – BitVector

מבוא

בתכנות מונחה עצמים (Object Oriented Programming) אנחנו מייצגים רעיונות באמצעות טיפוסי משתנים.
בחלק זה של העבודה נגדיר את הטיפוסים Bit ו – BitVector. המחלקה Bit מייצגת את הרעיון של ספרה בינארית. כלומר, אובייקט במחלקה מייצג ספרה שהערך שלה אחד או אפס. המחלקה BitVector מייצגת את הרעיון של מספר שלם (לא שלילי) המורכב מספרות בינאריות, ללא הגבלה על גודלו (בשונה מ – int ובדומה ל – BigInteger). בחלק זה של העבודה נייצג אובייקט מהטיפוס BitVector כמערך של אובייקטים מטיפוס Bit.

הנחיה: בתשובות לחלק זה של העבודה אין להשתמש בשיטות המחלקה BigInteger או כל מחלקה אחרת שלא אתם יצרתם ואשר מיועדת לייצוג מספרים גדולים.

המחלקה Bit

למחלקה Bit שדה פרטי אחד, value, מהטיפוס boolean. הספרה הבינארית "אפס" תיוצג על ידי עצם שהערך בשדה value שלו הוא false. הספרה הבינארית "אחד" תיוצג על ידי עצם שהערך בשדה value שלו הוא true.

במשימות להלן תכתבו בנאי ומספר שיטות של המחלקה Bit. אתם מקבלים קובץ בשם Bit.java שבו תבניות שאותן עליכם למלא.

```
public class Bit {
      private boolean value;
      public Bit(boolean value) { /* 4.1 הקוד הזה יושלם בסעיף */}
      public int toInt(){ /* 4.2 הקוד הזה יושלם בסעיף */}
      public String toString(){ /* 4.3 הקוד הזה יושלם בסעיף */}
}
```

משימה 4.1 (בנאי למחלקה Bit) (2 נקודות):

השלימו בקובץ Bit.java את הגדרת הבנאי:

```
public Bit(boolean value)
```

הבנאי מקבל כקלט ערך מטיפוס boolean.
אם הבנאי קיבל כארגומט את הערך false, הוא יוצר עצם המייצג את הספרה אפס. אחרת, הוא יוצר עצם המייצג את הספרה אחת. הבנאי לא זורק חריגות.

משימה 4.2 (החזרת הערך הדצימלי של העצם Bit) (2 נקודות):

השלימו בקובץ Bit.java את הגדרת הפונקציה:

```
public int toInt()
```

הפונקציה מחזירה את הערך 1 אם העצם מייצג את הספרה אחת. אחרת, יוחזר הערך 0. שיטה זו לא זורקת חריגות.

משימה 4.3 (החזרת המחרוזת של העצם Bit) (2 נקודות):

השלימו בקובץ Bit.java את הגדרת הפונקציה:

public string toString()

הפונקציה מחזירה את המחרוזת "1" אם העצם מייצג את הספרה אחת. אחרת, תוחזר המחרוזת "0". שיטה זו לא זורקת חריגות.

לדוגמא:

```
public static void main(String[] args) {
      Bit bit1 = new Bit(true);
      Bit bit0 = new Bit(false);
      System.out.println(bit0.toString() + " " + bit1.toString());
      int sum = bit1.toInt() + bit0.toInt();
      System.out.println(sum);
}
```

יודפס למסך:

0 1
1

המחלקה BitVector

המחלקה מייצגת מספרים גדולים כרצוננו ובפרט מספרים שגדולים מכדי להיות מיוצגים על ידי טיפוס פרימיטיבי.

למחלקה BitVector שדה פרטי אחד, bits, מטיפוס Bit[]. כל מספר (לא שלילי) מיוצג במחלקה זו על ידי המערך bits בייצוגו הבינארי. הייצוג הוא least significant bit first (lsbf). בייצוג זה הספרה הפחות משמעותית (זו שמייצגת את ה"אחדות") היא הביט הראשון במערך. מערך arr מטיפוס Bit[] מייצג מספר בינארי אם מתקיימים התנאים הבאים:

א. המערך arr אינו null ואינו ריק.
ב. כל האיברים במערך מייצגים ספרות בינאריות, כלומר הם עצמים מטיפוס Bit ואינם null.
ג. האיבר האחרון במערך (ה – msb) אינו מייצג את הספרה אפס, אלא רק במקרה שהמערך כולו מייצג את המספר אפס.

לדוגמא:

הייצוג הבינארי של המספר 13 הוא 1101. נוכל לייצר אובייקט מטיפוס BitVector שמייצג את המספר 13 באופן הבא (זכרו שהייצוג הוא lsbf): BitVector V = new BitVector("1011");

במשימות להלן תכתבנו בנאי ופונקציות של המחלקה BitVector. אתם מקבלים קובץ בשם BitVector.java שבו תבניות שאותן עליכם למלא.

הנחיה: בתשובות לחלק זה של העבודה אין להשתמש בשיטות המחלקה BigInteger או כל מחלקה אחרת שלא אתם יצרתם ואשר מיועדת לייצוג מספרים גדולים. מומלץ להשתמש בפונקציות הסטטיות שכתבתם בחלק 3 של עבודת הבית.

```
public class BitVector {
      private Bit[] bits;
      public BitVector(String s) { /* 4.4 הקוד הזה יושלם בסעיף */}
      public String toString(){ /* 4.5 הקוד הזה יושלם בסעיף */}
}
```

משימה 4.4 (בנאי למחלקה BitVector) (4 נקודות):

השלימו בקובץ BitVector.java את הגדרת הבנאי:

public BitVector(String s)

הבנאי מקבל כקלט מחרוזת s שהיא מחרוזת מספרית בבסיס 2. אין להניח דבר על הקלט. אם s אינה מחרוזת מספרית בבסיס 2, על הבנאי לזרוק חריגה מטיפוס IllegalArgumentException עם הודעה מתאימה לבחירתכם. אם הקלט תקין, הבנאי מאתחל את השדה bits כך שהאובייקט שנוצר מייצג את המספר ש – s מייצג.

משימה 4.5 (החזרת המחרוזת של העצם BitVector) (5 נקודות):

השלימו בקובץ BitVector.java את הגדרת הפונקציה:

public String toString()

הפונקציה מחזירה מחרוזת שמייצגת את הערך העשרוני של המספר שהאובייקט מייצג באופן שאנחנו רגילים לקרוא מספרים, כלומר בייצוג שהוא most significant bit first. שיטה זו לא זורקת חריגות.

לדוגמא:

```
public static void main(String[] args) {
      BitVector number = new BitVector("1011");
      System.out.println(number.toString());
}
```

יודפס למסך:
13

בהצלחה!