**<u>שאלה 1 (20 נק')</u>**

כתב אתב"ש (באנגלית A-Z,B-Y) הוא כתב סתרים המבוסס על חילופי אותיות מנוגדות באלפבית.
להלן פירוט הצופן עבור אותיות האלפבית האנגלי.
האותיות הגדולות מתחלפות כך:

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z | Y | X | W | V | U | T | S | R | Q | P | O | N |

כלומר, הקידוד של A הוא Z, הקידוד של Z הוא A, הקידוד של B הוא Y וכך הלאה. לדוגמא, המלה ABC תקודד ע"י מילת הצופן ZYX .
בשאלה זו, כל התוים שאינם אותיות אנגליות <u>גדולות</u> (למשל: '-', '^', 'a') אינם מוצפנים, כלומר נותרים כפי שהם.

למשל, התוכנית הבאה קוראת לשתי פונקציות, encrypt ו- decrypt, ומדפיסה את הפלטים של הפונקציות הנ"ל:

```
public static void main(String[] args) {
	String str = "BEN-GURION-RULES!!!";
	String encrypted = encrypt(str);
	System.out.println("encrypted: " + encrypted);
	String decrypted = decrypt(encrypted);
	System.out.println("decrypted: " + decrypted);
}
```

למסך יודפס הפלט:

encrypted: YVM-TFIRLM-IFOVH!!!

decrypted: BEN-GURION-RULES!!!

(המשך השאלה בעמוד הבא)

סעיף א' (15 נק'):

השלימו את הפונקציה encrypt המקבלת מחרוזת ומצפינה אותה באמצעות הצופן המתואר לעיל.

```
public static String encrypt(String str) {
      String encrypted = "";
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
      return encrypted;
}
```

סעיף ב' (5 נק'):

השלימו את הפונקציה decrypt המקבלת מחרוזת שהוצפנה באמצעות הצופן המתואר לעיל ומפענחת אותה. בסעיף זה, פתרון ארוך מארבע שורות לא יתקבל.

```
public static String decrypt(String str) {
```

_____

_____

_____

_____

```
}
```

**<u>שאלה 2 (20 נק')</u>**

נתונה השיטה הבאה, במחלקה List:

```
public class List {
	public Link head;

	public Object what(List list1, List list2){
		Link head1 = list1.head, head2 = list2.head, temp1,temp2;
		int i1, i2, i;
		Object ans = null;
		for(temp1=head1, i1=0; temp1!=null; temp1=temp1.next, i1=i1+1);
		for(temp2=head2, i2=0; temp2!=null; temp2=temp2.next, i2=i2+1);
		if (i1<i2){
			temp1=head1;
			head1=head2;
			head2=temp1;
			i=i1;
			i1=i2;
			i2=i;
		}

		for (i1=i1-i2;i1>0;i1=i1-1)
			head1=head1.next;
		while (head1 != null & ans == null){
			if (head1==head2) ans = head1.data;
			head1=head1.next;
			head2=head2.next;
		}
		return ans;
	}
}
```

<u>סעיף א' (11 נקודות)</u>

תארו בשורה אחת את יעודה של השיטה what.

---

<u>סעיף ב' (9 נקודות)</u>

עבור כל אחד מהסעיפים הבאים צייר את list1 **ואת** list2, הניתנים כקלט לפונקציה what, על מנת לקבל את הפלט המצוין. במידה ויש יותר מאפשרות אחת, ציירו את אחת האפשרויות.
במידה ואין אף אפשרות, כיתבו "בלתי אפשרי" והסבירו בקצרה מדוע.

1) הרשימה list2 נתונה ואיבריה הם [4]→[5]→[5]→[3]→[2]→[3], השיטה what(list1,list2) החזירה 3, ונתון כי אורך הרשימה list1 הוא 3.

2) הרשימה list2 נתונה ואיבריה הם [9]→[2]→[5]→[6]→[2]→[12]→[7], השיטה what(list1,list2) החזירה 2, ונתון כי אורך הרשימה list1 הוא 4.

3) הרשימה list1 נתונה ואיבריה הם [6]→[5]→[4]→[3]→[2]→[1], השיטה what(list1,list2) החזירה 3, ונתון כי אורך הרשימה list2 הוא 7.

**<u>שאלה 3 (20 נק')</u>**

סדרת מספרים חיובית-יורדת הינה סדרה בה כל מספר הינו קטן יותר מהמספר הקודם לו בסדרה, וכל איברי הסדרה הינם חיוביים (גדולים מ 0).
למשל, הסדרות הבאות הינן חיוביות יורדות (משמאל לימין):

6 4
6 5 2 1

בעוד שהסדרות הבאות אינן חיוביות יורדות:

2 4
5 5 3

השלימו את הפונקציה הרקורסיבית allDecreasing(n,k) המדפיסה את כל הסדרות החיוביות-יורדות באורך k בהן המספר הגדול ביותר הינו n. פונקציה זו קוראת לפונקצית עזר רקורסיבית.

למשל, הקריאה allDecreasing(4,3) תדפיס למסך:

3 2 1
4 2 1
4 3 1
4 3 2

יש להפריד בין איברי הסדרה ע"י רווח.
אין חשיבות לסדר ההדפסה של הסדרות. ניתן להניח כי k,n אינם שליליים.

```
public static  void  allDecreasing (int n, int k) {

        allDecreasing(_____);

}

private static void allDecreasing (_____) {

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

}
```

**<u>שאלה 4 ( 25 נק')</u>**

בשאלה זו נשנה את המחלקה BNode שנלמדה בכתה, כך שתכיל מצביע גם לצומת ה"אבא":

```
public class BNode {
      public Object data;
      public BNode left;
      public BNode right;
      public BNode parent;
      ...
}
```

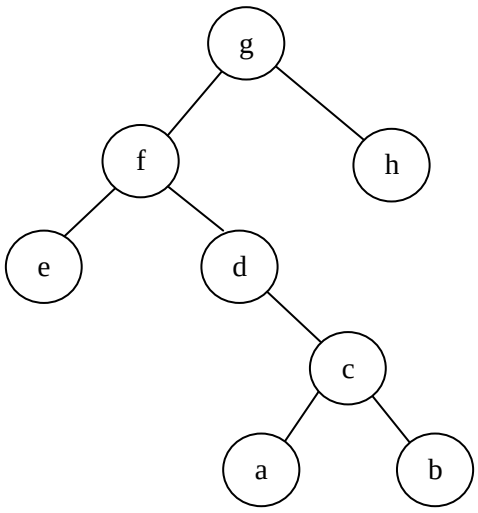
בדומה למימוש שנלמד בכתה, במידה ואין לקודקוד בן שמאלי אזי left הוא null ובמקרה בו אין בן ימני אזי right הוא null. במימוש זה במקרה בו אין "אבא" (כלומר לשורש), parent הוא null.

<u>סעיף א' (3 נק')</u>

הוסיפו למחלקה BNode את השיטה ( )leftMost המחזירה את הצומת השמאלי ביותר בתת העץ של עצם המפתח this (כלומר הצומת שהיה מודפס ראשון בסריקת InOrder).

<u>תזכורת</u>: בסריקת InOrder, עבור כל צומת בעץ, יש לעבור תחילה על כל הצמתים בתת העץ השמאלי של הצומת, לאחר מכן להגיע לצומת, ולבסוף לעבור על כל הצמתים בתת העץ הימני של הצומת.

למשל, עבור העץ הנתון משמאל:




הקריאה ( )g.leftMost תחזיר את הצומת e
הקריאה ( )d.leftMost תחזיר את הצומת d
הקריאה ( )c.leftMost תחזיר את הצומת a

```
public BNode leftMost (){

      BNode result;
```

_____

_____

_____

_____

_____

```
}
```

<u>סעיף ב' (7 נק')</u>

נאמר כי קודקוד x הוא "צאצא שמאלי" של קודקוד y אם x נמצא בתת העץ השמאלי של y.
הוסיפו למחלקה BNode את השיטה ( )lowAncestor המחזירה את האב הקדמון הנמוך ביותר של עצם המפתח (this) כך שעצם המפתח הינו צאצא שמאלי שלו, או null אם לא קיים אב קדמון כזה.
למשל עבור העץ בעמוד הקודם:

| | |
|---|---|
| הקריאה ( )b.lowAncestor, תחזיר את הצומת g | הקריאה ( )c.lowAncestor, תחזיר את הצומת g |
| הקריאה ( )d.lowAncestor, תחזיר את הצומת g | הקריאה ( )a.lowAncestor, תחזיר את הצומת c |
| הקריאה ( )e.lowAncestor, תחזיר את הצומת f | הקריאה ( )g.lowAncestor, תחזיר null |
| הקריאה ( )h.lowAncestor, תחזיר null | |

```
public  BNode lowAncestor() {
      BNode result = this;
```

_____

_____

_____

_____

_____

_____

_____

```
}
```

<u>סעיפים ג',ד' (5 + 10 נק')</u>

בסעיפים הבאים נשנה את המחלקה BTree המייצגת עץ בינארי כך שניתן יהיה לעבור על הערכים בצמתי העץ בזה אחר זה באופן סדרתי. לשם כך המחלקה BTree תממש את הממשק Iterable:

```
public class BTree implements Iterable{
     private BNode root;
     ...
     public Iterator iterator(){
           return new TreeIterator();
     }
}
```

הוספנו מחלקה פנימית פרטית בתוך BTree הממשת את הממשק Iterator ושמה TreeIterator, המאפשרת לעבור על צמתי העץ באופן סדרתי לפי סדר InOrder.
השלימו את השיטה ( )hasNext הבודקת האם קיים צומת שעוד לא נסרק (5 נק')
ואת השיטה ( )next המחזירה את הצומת הבא ע"פ סריקת InOrder (10 נק')

<u>הערה</u>: מחלקה פנימית היא מחלקה המוגדרת בתוך מחלקה אחרת. התועלת בכך היא שלמחלקה הפנימית יש גישה לחלקים הפרטיים של המחלקה בתוכה היא מוגדרת ואין צורך להפוך אותם לציבוריים.
<u>תזכורת</u>:

```
interface Iterable {
      public Iterator iterator();
}
interface Iterator {
      public boolean hasNext();
      public Object next();
}
```

לדוגמה, עבור BTree t העץ שתואר בציור בתחילת השאלה, הרצת הקוד הבא תדפיס את ערכי הצמתים לפי סריקת InOrder:

```
Iterator it = t.iterator();
while (it.hasNext()) {
      System.out.print(it.next()+ " ");
}
```

==========

```
public class BTree implements Iterable{
     private BNode root;
     public Iterator iterator() {return new TreeIterator();}
     ...
     private class TreeIterator implements Iterator{
           private BNode current;
           public  TreeIterator(){
                if (root != null)
                   current = root.leftMost(); // קריאה לשיטה מסעיף א'
                else
                   current = null;
           }
           public boolean hasNext(){

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

          }

          public Object next(){
            Object result = current.getData();
            BNode nextNode;
             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

             _____________________________________________________

            current = nextNode;
            return result;
          }
     }
}
```

**<u>שאלה 5 (15 נק')</u>**

התבוננו במחלקות A, B ו-AException המופיעות בעמודים הבאים ובקטע הקוד הבא:

```
public static void main(String [] args) {
        A aa = new A(0,1);
        A ab = new B(0,1);
        B bb = new B(-2,-1);

        // הוספת השורה בכל סעיף הינה במקום זה
}
```

לגבי כל אחת משורות הקוד הבאות (1-15), סמנו מה יקרה אם נכתוב אותה במקום שורת ההערה ב-main – (נק' אחת לכל סעיף)

א. תהיה שגיאת קומפילציה. במקרה זה יש גם לציין את **הסיבה לשגיאה**.

ב. תהיה שגיאה בזמן ריצה (התוכנית "עפה"). במקרה זה יש לציין את **הסיבה לשגיאה (חריגה)**.

ג. הקוד ירוץ בצורה תקינה. במקרה זה יש לכתוב **את הפלט שיודפס למסך**.

תתכן הדפסה של עד שתי שורות.

<u>שימו לב</u>: קוד המכיל שגיאה (חריגה) שנתפסת ע"י catch נחשב לקוד שרץ בצורה תקינה.

**1)** System.out.println(ab.a);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**2)** System.out.println(bb.a);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**3)** System.out.println(bb.b);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**4)** System.out.println(aa.getB());

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**5)** System.out.println(((B)aa).getB());

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**6)** System.out.println(ab.getB());

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**7)** System.out.println(((A)ab).getB());

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**8)** System.out.println(aa);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**9)** System.out.println(ab);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**10)** System.out.println(aa.f());

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**11)** System.out.println(aa.f(1));

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**12)** System.out.println(ab.f());

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**13)** System.out.println(ab.f(-1));

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**14)** System.out.println(bb.f());

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

**15)** System.out.println(bb.f(-1));

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה/הפלט:_____

_____

```
public class A {
    public int a;
    private int b;

    public A(int a, int b) {
        this.a = a;
        this.b = b;
    }

    public int getB() {
        return b;
    }

    public int f() {
        return f(0);
    }

    public int f(int c) {
        if (c <= 0)
            throw new AException("c was "+c);
        else
            return b+c;
    }

    public String toString() {
        return "a="+a+",b="+b;
    }
}
public class B extends A {
    public int a;
    private int b;

    public B(int a, int b) {
        super(a,b);
        this.a = a+1;
        this.b = b+1;
    }

    public int getB() {
        return b;
    }

    public int f() {
        int c = 0;
        try {
            c = super.f();
        }
        catch (AException e){
            System.out.println("BException: "+e.getMessage());
        }
        return c;
    }

    public int f(int c) {
        return super.f(b+c);
    }

    public String toString() {
        return "a="+a+",b="+b;
    }
}
public class AException extends RuntimeException {
    public AException(String errMsg) {
        super("AException: "+errMsg);
    }
}
```

**תזכורת:**

• int **length**()

שיטה במחלקה String המחזירה את אורך המחרוזת

• int **indexOf**(char c)

שיטה במחלקה String המחזירה את האינדקס הראשון במחרוזת, בו מופיע התו c,
או 1- אם c לא מופיע כלל במחרוזת

• char **charAt**(int index)

שיטה במחלקה String המחזירה את התו במיקום index במחרוזת

• String **substring**(int beginIndex, int endIndex)

שיטה במחלקה String, המחזירה מחרוזת חדשה שהיא תת מחרוזת של עצם המפתח, החל מהמיקום ה-
beginIndex (כולל) ועד המיקום endIndex (לא כולל).

• double **Math.pow**(double a, double b)

פונקציה המחזירה את הערך של a בחזקת b

• double **Math.sqrt**(double a)

פונקציה המחזירה את השורש הריבועי החיובי של a

• double **Math.abs**(double a)

פונקציה המחזירה את ערכו המוחלט של a

• int **Math.abs**(int a)

פונקציה המחזירה את ערכו המוחלט של a

• double **Math.min**(double a, double b)

פונקציה המחזירה את הערך המינימלי מבין b, a

• int **Math.min**(int a, int b)

פונקציה המחזירה את הערך המינימלי מבין b, a

• double **Math.max**(double a, double b)

פונקציה המחזירה את הערך המקסימלי מבין b , a

• int **Math.max**(int a, int b)

פונקציה המחזירה את הערך המקסימלי מבין b , a