מבוא למדעי המחשב
202-1-1011

סמסטר א' תש"ע
מבחן מועד ב'

מבוא לתכנות למערכות מידע
202-1-1041

מר טל גרינשפון
פרופ' משה זיפר
מר שי זקוב
ד"ר פז כרמי
פרופ' מיכאל קודיש
גב' אירינה רבייב
ד"ר צחי רוזן

משך המבחן: שלוש וחצי שעות.
חומר עזר אסור.

במבחן זה 6 שאלות בניקוד המסתכם ל-110 נקודות. עליכם לסמן בדף התשובות שאלה/סעיף/סעיפים שסכומם הכולל הוא 10 נקודות עליהם בחרתם לא לענות. לדוגמה, ניתן לסמן את סעיפים א' ב' וג' משאלה 1, או את סעיף א' משאלה 2. **במידה ולא תסמנו שאלה/סעיף/סעיפים עליהם בחרתם לא לענות או סכום הכולל של שאלה/סעיף/סעיפים שבחרתם שונה מ-10 נקודות , הבדיקה לא תיקח בחשבון את שאלה 1 .**

**אנא רשמו את תשובותיכם בטופס התשובות בלבד.** המחברת שקיבלתם הנה מחברת טיוטה בלבד ולא תימסר כלל לבדיקה. הקפידו לרשום על טופס הבחינה גם את מספר הנבחן וגם את מספר החדר בו אתם נבחנים. מספר השורות העומדות לרשותכם בגוף השאלון רומז על אורך התשובה הנדרשת. הקפידו על כתב יד ברור. תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא.

**בכל שאלה מותר להסתמך על סעיפים אחרים של אותה השאלה גם אם לא עניתם עליהם.**

**אם לשאלה מצורף שלד של המחלקה, מותר להשתמש אך ורק בשיטות ושדות שצוינו בפירוש בתוך השלד הנתון בשאלה, או בשיטות מסעיפים אחרים של אותה השאלה.**

בשאלות בהן לא מצוין אחרת, ניתן לבחור בפתרון רקורסיבי או פתרון שאינו רקורסיבי, לבחירתכם. **שימו לב:** החשיבות העליונה היא לנכונות הקוד. מאידך, יעילות וסגנון חשובים גם הם, ולכן יורדו נקודות על תשובות אשר כוללות קוד לא יעיל או שאינו כתוב בהתאם לכללי הסגנון שנלמדו (סגנון: אינדנטציה, שמות משתנים, שיטות ומחלקות, הוספת הערות לגבי תפקידם של משתנים נוספים עליהם הצהרתם).

**במידה ואינכם יודעים את התשובה לסעיף כלשהו, רשמו "לא יודעים" (במקום תשובה) ותזכו ב-20% מניקוד הסעיף. לא ניתן לכתוב לא יודע על חלק מסעיף.**

# בהצלחה!!!

## **שאלה 1 (10 נק')**

לעץ בינארי T ביצעו סריקת PostOrder, שתוצאתה (משמאל לימין): a,b,c,d,e,f,g

**סעיף א** (3 נקודות)

מהו ערכו של שורש העץ T?

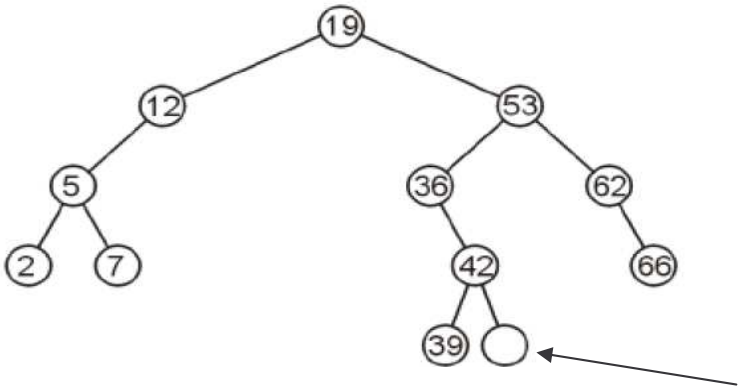
**סעיף ב** (3 נקודות)

נניח שתת העץ הימני של T מכיל את הערכים d,e,f.

מהו ערכו של שורש תת-העץ השמאלי של T?

**סעיף ג** (4 נקודות)

מהו טווח המפתחות (הערכים) שניתן להכניס במקום הקודקוד הריק בעץ החיפוש הבינארי הבא (הניחו שכל המפתחות הם מספרים שלמים **ללא חזרות**):



## שאלה 2 (20 נק')

נתון הממשק הבא המגדיר פוליגון (הממשק זהה לזה שהוגדר בתרגיל 4).

```
public interface Polygon {
        public Polygon cloneMe();
        public double perimeter();
        public double area();
        public Point[] getVertices();
        public void translate (Point p);
        public boolean contains(Point p);
        public boolean contains(Polygon other);
        public void scale (Point p, double scaleFactor);
}
```

יש להוסיף מחלקה AbsPolygon מופשטת (abstract) על מנת לחסוך קוד ולשפר את הממשק.
על המחלקה AbsPolygon לממש את ממשק Polygon. המחלקות Triangle ו- Rectangle וגם מחלקות נוספות המתארות מצולעים שונים ירשות מהמחלקה AbsPolygon.

בסעיפים הבאים אתם מתבקשים להשלים את השיטות contains(Polyogon other)
ו-equalsPolygon (Polygon other) בתוך מחלקה AbsPolygon.
ניתן להניח שבפוליגון אין שתי נקודות זהות.

### סעיף א' (10 נק')

השלימו את השיטה contains בתוך המחלקה AbsPolygon.

```
public boolean contains(Polygon other){

        השלימו בדף התשובות//
}
```

השיטה contains בודקת האם other מוכל כולו בתוך הפוליגון הנוכחי (this). מותר להניח ש-other שונה מ-null.

### סעיף ב' (10 נק')

השלימו את השיטה equalsPolygon בתוך המחלקה AbsPolygon.

```
public boolean equalsPolygon (Polygon other){
        השלימו בדף התשובות//
}
```

השיטה equalsPolygon מחזירה אמת אם ורק אם other מוגדר מאותן הנקודות כמו הפוליגון הנוכחי (this).

## שאלה 3 (20 נקודות)

**קראו את השאלה עד הסוף וראו את דוגמת הרצה בעמוד הבא בטרם תתחילו לפתור את השאלה!**

נתון הממשק BankAccount המגדיר חשבון בנק שהמשיכה ממנו מוגבלת בצורה כלשהי.
למשל, חשבון חיסכון שלא ניתן למשוך ממנו כסף כלל, או חשבון עם-משיכת-יתר-מוגבלת שלא ניתן לחרוג בו מעבר להגבלה, או חשבון ללא-משיכת-יתר שלא ניתן למשוך ממנו מעבר ליתרה.

```
public interface BankAccount {
      public void deposit(double amount);   // הפקדה
      public double withdraw(double amount);// משיכה
      public double balance();              // יתרה
}
```

הפקדה לחשבון נעשית באמצעות השיטה **deposit**(amount) המוסיפה את amount ליתרה.
משיכה מהחשבון נעשית באמצעות השיטה **withdraw**(amount). השיטה פועלת באופן הבא:
- בודקת מה סכום המשיכה המקסימאלי האפשרי כרגע (המחושב על-פי היתרה הנוכחית ומסגרת האשראי)
- אם הסכום המקסימאלי למשיכה גדול מ- amount , אז מוחזר amount,
- אחרת מוחזר הסכום המקסימאלי למשיכה.
השיטה מעדכנת את היתרה הנוכחית בחשבון בהתאם למשיכה.

**שימו לב**: ניתן להניח שבשתי השיטות deposit ו- withdraw הפרמטר amount חיובי ואין צורך לבדוק זאת.

השיטה **balance**() מחזירה את היתרה הנוכחית בחשבון.

השלימו את שלושת המחלקות הבאות כמפורט.
**עליכם לכתוב את המחלקות בצורה החסכונית ביותר. כלומר, לממש את השיטות במחלקה הכי גבוהה בהיררכיה שניתן וטבעי לממש בה את השיטות. אין צורך לממש את השיטות toString ו- equals.**

1. מחלקה Account - מחלקה מופשטת (abstract) העומדת בראש היררכיה של קבוצת המחלקות המגדירות חשבונות בנק שהמשיכה מהם מוגבלת בצורה כלשהי. המחלקה ממשת את הממשק BankAccount.

   המחלקה מגדירה בנאי המקבל את היתרה הראשונית בחשבון.

2. מחלקה LimitedAccount.

   מחלקה המגדירה חשבון בנק עם יתרה מוגבלת. כלומר, לא ניתן לבצע משיכת יתר מעבר להגבלה, במידה ולא ניתן למשוך את הסכום המבוקש מושכים את הסכום המקסימאלי הניתן.
   המחלקה היא סוג של Account.
   המחלקה מגדירה בנאי המקבל את היתרה הראשונית בחשבון ואת הסכום המותר למשיכת יתר.
   בנוסף, המחלקה מגדירה שיטה ציבורית **double limit**() המחזירה את הסכום המותר למשיכת יתר.
   המחלקה מגדירה שיטה ציבורית **improve**(double amount) המחזירה LimitedAccount חשבון חדש משופר בו הסכום המותר למשיכת יתר גדל ב- amount (ביחס לעצם this).

3. המחלקה NoOverdraftAccount.

   מחלקה המגדירה חשבון בנק ללא משיכת יתר.
   המחלקה היא סוג של LimitedAccount.
   המחלקה מגדירה בנאי המקבל את היתרה הראשונית בחשבון.

להלן תוכנית דוגמה (הפלט רשום כהערות).

```
public static void main(String[] args) {
      BankAccount ba1 = new LimitedAccount(100, 100);
      System.out.println("(" + ba1 + ", " + ba1.balance() + ", "
                  + ((LimitedAccount) ba1).limit() + ")");
      // (LimitedAccount@c17164, 100.0, 100.0)

      ba1.deposit(100);
      System.out.println("(" + ba1 + ", " + ba1.balance() + ", "
                  + ((LimitedAccount) ba1).limit() + ")");
      // (LimitedAccount@c17164, 200.0, 100.0)

      System.out.println(ba1.withdraw(250));
      // 250.0

      System.out.println("(" + ba1 + ", " + ba1.balance() + ", "
                  + ((LimitedAccount) ba1).limit() + ")");
      // (LimitedAccount@c17164, -50.0, 100.0)

      System.out.println(ba1.withdraw(100));
      // 50.0

      System.out.println("(" + ba1 + ", " + ba1.balance() + ", "
                  + ((LimitedAccount) ba1).limit() + ")");
      // (LimitedAccount@c17164, -100.0, 100.0)

      BankAccount ba2 = new NoOverdraftAccount(100);
      System.out.println("(" + ba2 + ", " + ba2.balance() + ")");
      // (NoOverdraftAccount@61de33, 100.0)

      ba2.deposit(100);
      System.out.println("(" + ba2 + ", " + ba2.balance() + ")");
      // (NoOverdraftAccount@61de33, 200.0)

      System.out.println(ba2.withdraw(150));
      // 150.0

      System.out.println("(" + ba2 + ", " + ba2.balance() + ")");
      // (NoOverdraftAccount@61de33, 50.0)

      System.out.println(ba2.withdraw(100));
      // 50.0

      System.out.println("(" + ba2 + ", " + ba2.balance() + ")");
      // (NoOverdraftAccount@61de33, 0.0)

      ba3 = ((LimitedAccount) ba1).improve(100);
      System.out.println("(" + ba3 + ", " + ba3.balance() + ", "
                  + ((LimitedAccount) ba3).limit() + ")");
      // (LimitedAccount@a18264, -100.0, 200.0)

      ba4 = ((LimitedAccount) ba2).improve(100);
      System.out.println("(" + ba4 + ", " + ba4.balance() + ", "
                  + ((LimitedAccount) ba4).limit() + ")");
      // (LimitedAccount@14318bb, 0.0, 100.0)
}
```

## שאלה 4 (20 נקודות)

### סעיף א' (10 נק')

השלימו את קוד הפונקציה הבאה, אשר ממיינת (בסדר עולה) מערך אובייקטים. הפונקציה מקבלת כקלט מערך אובייקטים ומצביע ל-Comparator, המגדיר את קריטריון הסידור. רשמו בדף התשובות את שם אלגוריתם המיון בו אתם משתמשים (אתם רשאים להשתמש בכל אחד מאלגוריתמי המיון שנלמדו בכיתה או בתרגול). תזכורת לממשק Comparator ניתן למצוא בהמשך.

```
public static void sort(Object[] arr, Comparator comp){
        //השלימו בדף התשובות
}
```

### סעיף ב' (10 נק')

להלן פרוצדורה המבצעת העמסה (Overloading) על הפרוצדורה מהסעיף הקודם:

```
public static void sort(Comparable[] arr){
        sort(arr, new MyComparator());
}
```

השלימו את קוד המחלקה MyComparator, כך שהפרוצדורה הנתונה תמיין נכון מערך של אוביקטים מסוג Comparable. תזכורת לממשק Comparable ניתן למצוא בהמשך.

```
public class MyComparator implements Comparator {
        //השלימו בדף התשובות
}
```

תזכורת לממשקים Comparator ו-Comparable:

```
public interface Comparator {
        /**
         * Decides the order between two compared objects.
         * @param o1 the first object.
         * @param o2 the second object.
         * @return  a negative value, 0, or a positive
         * value, if the rank in the order of the first
         * object is lower, identical, or higher than the
         * rank of the second object, respectively.
         */
        public int compare(Object o1, Object o2);
}

public interface Comparable {
        /**
         * Decides the order between this object and
         * another given object.
         * @param other an object to compare with this
         * Comparable object
         * @return a negative value, 0, or a positive value,
         * if this object has a lower, an identical, or a
         * higher rank in the order, respectively.
         */
        public int compareTo(Object other);
```

}

## שאלה 5 ( 20 נק')

נתונה המחלקה List המתוארת כאן (ללא מימוש השיטות). אין להשתמש בשיטות או שדות מלבד אילו המופיעים כאן ולא ניתן להוסיף שיטות עזר.

```
public class List {
        /**   Constructor: creates a new empty list.*/
        public List() { . . . }

        /** Return true, if and only if the list is empty */
        public boolean isEmpty(){ . . . }

        /**  Adds a given object to the front of the list */
        public void addFirst(Object toAdd) { . . . }

        /** Removes an object from the front of  the list. Returns null if the list is empty */
        public Object removeFirst(){ . . . }

        /**  Checks if a given object is a member of the list */
        public boolean isMember(Object obj) { . . . }

        /**  reverse the order of the items in the list */
        public void reverse(){ . . . }
}
```

### סעיף א'  (10 נק')

הוסיפו למחלקה List שיטה בשם split המקבלת כפרמטר אובייקט obj, מסירה את כל אברי הרשימה עד למופע הראשון (כולל) של האובייקט obj ומחזירה רשימה חדשה המכילה את כל האיברים שהוסרו (בסדר המקורי בו הופיעו). במידה והאובייקט אינו מופיע ברשימה, על השיטה להחזיר רשימה ריקה (המכילה אפס איברים). ניתן להניח ש-obj שונה מ-null. ניתן להשתמש ב- equals על מנת לבדוק שיוון לוגי בין אוביקטים.

```
public List split(Object obj){
                        // השלימו את הקוד בדף התשובות
}
```

דוגמא 1:

```
List list = new List();
list.addFirst(123);
list.addFirst(453);
list.addFirst(12);
list.addFirst(453);
list.addFirst(76);
list.addFirst(8);

System.out.println(list); //print: [8,76,453,12,453,123]
System.out.println(list.split(12)); //print: [8,76,453,12]
System.out.println(list); //print [453,123]
```

דוגמא 2:

```
List list = new List();
list.addFirst(123);
list.addFirst(453);
list.addFirst(12);
list.addFirst(453);
list.addFirst(76);
list.addFirst(8);

System.out.println(list); //print: [8,76,453,12,453,123]
System.out.println(list.split(128)); //print: []
System.out.println(list); //print [8,76,453,12,453,123]
```

## סעיף ב' (10 נק')

כתבו את השיטה הבאה במחלקה List

**public void removeAllOccurrences(Object key)**

על השיטה לקבל כפרמטר אובייקט key, ולשנות את הרשימה באופן הבא: על השיטה להסיר את כל החוליות ששדה ה- data שלהן שווה ל- key. אם האובייקט key אינו מופיע ברשימה, אז לא מתרחש שינוי. ניתן להניח ששדות ה-data ברשימה אינם null.

לדוגמה, נתונה רשימה מקושרת

התוצאה אחרי הפעלת השיטה removeAllOccurrences עם key = 'b':

## **שאלה 6 (20 נקודות)**

בהינתן שתי מחרוזות s1 ו-s2, נאמר ש-s2 היא **תת –קבוצה סדורה** של s1 אם s1 מכילה את כל התווים של s2 באותו הסדר בו הם מופיעים ב-s1. לדוגמה, מחרוזת "abc" היא תת-קבוצה סדורה של "xaybcde", אבל אינה תת-קבוצה סדורה של "xbyacde" כיוון שתווים 'a' ו-'b' אינם מופיעים באותו הסדר במחרוזות "abc" ו-"xbyacde").

בשאלה זו הנכם רשאים להשתמש רק בשיטות charAt() ו- length() של המחלקה String.
תזכורת: השיטה charAt(int i) מחזירה את התו באינדקס i של המחרוזת.
השיטה length() מחזירה את אורך המחרוזת.

כתבו פונקציה רקורסיבית בשם orderedSubsetsOfSize המקבלת מחרוזת s ומספר שלם k ומדפיסה את כל תתי-הקבוצות הסדורות באורך k **בהן לא מופיעים שני תווים עוקבים של s.**

למשל, הפונקציה main הנתונה קוראת ל- orderedSubsetsOfSize(s, k) עם ארגומנטים s = "xaybcd" ו- k=3, והתכנית תדפיס את הקבוצות הסדורות הבאות (אחת לשורה בסדר כלשהו בין הקבוצות):

abd
xbd
xyd
xyc

**הניחו כי:**
**1) s אינה null ומכילה תווים שונים**
**2) $k \geq 0$.**

```
class OrderedSubsets{

    public static void orderedSubsetsOfSize(String s, int k){
        orderedSubsetsOfSize(s,k,0,"");
     }

    public static void orderedSubsetsOfSize(String s, int k,int index, String orderedSubstr) {
        //השלימו בדף התשובות
    }



   public static void main(String[] args){
        orderedSubsetsOfSize("xaybcd",3);
   }
 }
```