מבוא למדעי המחשב
202-1-1011

סמסטר א' תש"ע
מבחן מועד א'

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 שאלה 6, או את סעיפים ב' וג' משאלה 3, או את סעיף א' משאלה 3. **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 6.**

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 שצוינו בפירוש בתוך השלד, או שיטות 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 (20 נק')

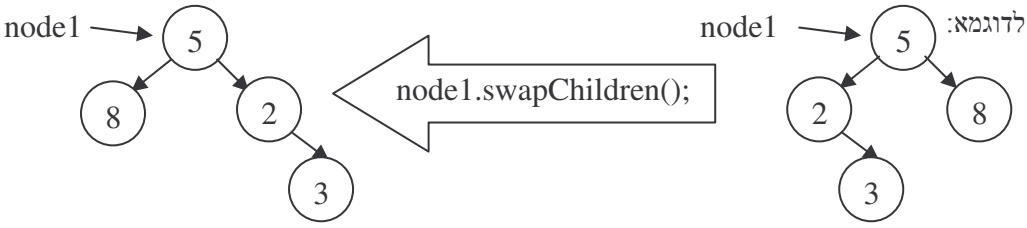
נתונות המחלקות BinaryNode ו-BinaryTree:

```
public class BinaryNode {
        protected Object data;
        protected BinaryNode left;
        protected BinaryNode right;
…
}

public class BinaryTree {
        private BinaryNode root;
…
}
```

בדוגמאות שדה ה- data בתוך BinaryNode הוא מטיפוס Integer.

סעיף א' (5 נקודות)

עליכם לממש את השיטה swapChildren בתוך המחלקה BinaryNode, וכתוצאה מהפעלת השיטה, מוחלפים שני הצאצאים של הצומת.



מבנה השיטה:

```
public void swapChildren(){
        //      השלימו שיטה זו בדף התשובות
}
```

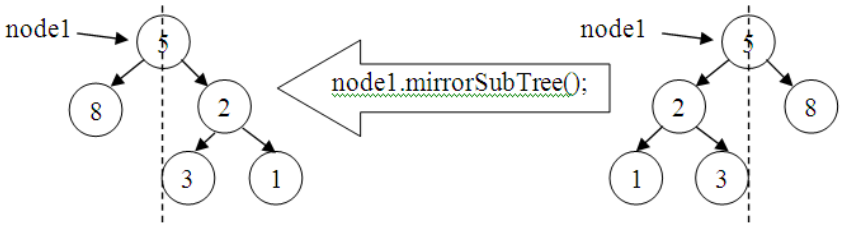
<u>סעיף ב' (10 נקודות)</u>

שיקוף של עץ T מוגדר באופן הבא:

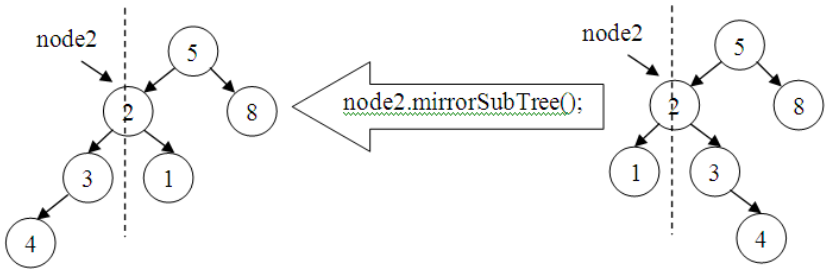
- אם T עץ ריק, אז השיקוף של T הינו עץ ריק.
- אחרת , שיקוף של T הוא עץ המתקבל מהחלפת הבנים של שורש העץ T, ושיקוף תת העץ השמאלי ותת העץ הימני של T.

עליכם לממש את השיטה **<u>הרקורסיבית</u>** mirrorSubTree בתוך המחלקה BinaryNode. כתוצאה מהפעלת השיטה, תת העץ המושרש בצומת עליו הופעלה השיטה עובר שיקוף.

<u>דוגמה 1:</u>



<u>דוגמה 2:</u>

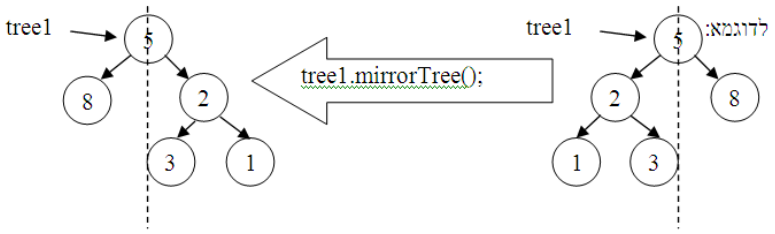


מבנה השיטה:

```
public void mirrorSubTree(){
        //          השלימו שיטה זו בדף התשובות
}
```

<u>סעיף ג' (5 נקודות)</u>

עליכם לממש את השיטה mirrorTree בתוך המחלקה **BinaryTree**. כתוצאה מהפעלת השיטה, העץ עליו הופעלה השיטה עובר שיקוף מהשורש.



מבנה השיטה:

```
public void mirrorTree(){
        //          השלימו שיטה זו בדף התשובות
}
```

שאלה 2 (20 נק')

נתון הממשק הבא לתור

```
public  interface Queue{
   public boolean isEmpty(); // returns true if and only if the queue is empty
   public Object dequeue(); // removes and returns the first item in the queue
   public void enqueue(Object o); // inserts an item at the end of the queue
}
```

ונתונות שתי פונקציות סטאטיות הבאות:

```
public static boolean mystery( Queue q1, Queue q2 ){
        if( q1.isEmpty() & q2.isEmpty() )
                return true;
        if(q1.isEmpty() | q2.isEmpty())
                return false;
        if( ! (q1.dequeue().equals( q2.dequeue())) )
                return false;
       return mystery( q1, q2 );
}

public static void secret( Queue q ){
        if( !q.isEmpty() ){
                Object obj = q.dequeue();
                secret( q );
                q.enqueue( obj );
        }
}
```

ענו בדף התשובות על הסעיפים הבאים

סעיף א' (5 נקודות)

רשמו במשפט אחד מה מבצעת שיטה mystery.

סעיף ב' (10 נקודות)

רשמו במשפט אחד מה מבצעת שיטה secret.

סעיף ג' (5 נקודות)

תנו דוגמא לזוג תורים q1, q2 כך שבכל תור ישנם בין 3 ל 5 איברים שונים מטיפוס Integer, עבורם רצף הפקודות הבא ידפיס את הערך true:

```
secret(q2);
System.out.println(mystery(q2, q1));
```

שאלה 3 (20 נקודות)

נתון הממשק Function המגדיר פונקציה חד-מקומית y = f(x) כלהלן:

```
public interface Function {
  public double valueAt (double x);
}
```

השיטה **valueAt(x)** מחזירה את הערך של f בנקודה x. למשל, אם הפונקציה f היא y = 2x + 3 ו- x = 5 , אז השיטה מחזירה ערך 13.

עליכם לכתוב בדף התשבות את אוסף המחלקות המתוארות בסעיפים א'-ג'.
קראו את השאלה עד הסוף בטרם תתחילו לפתור אותה!

<u>שימו לב:</u> יש לכתוב את המחלקות בצורה החסכונית ביותר. **כלומר, לממש את השיטות במחלקה הכי גבוהה בהיררכיה שניתן וטבעי לממש בה את השיטות.**

על המחלקות לתמוך בשיטת toString() (בין אם באמצעות מימוש או הורשה) כך שהשיטה *String* ***toString()*** מחזירה מחרוזת המתארת את הפונקציה. למשל המחרוזת "y = 2x + 3" תתאר את הפונקציה y = 2x + 3.

<u>סעיף א' (10 נקודות)</u>

המחלקה DerivedFunction הינה מחלקה מופשטת (abstract) העומדת בראש ההיררכיה של קבוצת המחלקות המממשות קבוצה של פונקציות חד-מקומיות גזירות. המחלקה ממשת את הממשק Function.

בנוסף, המחלקה מגדירה שתי שיטות ציבוריות.

public double square(double x) המחזירה את הערך של הפונקציה $y = f^2(x)$ בנקודה x. למשל, אם הפונקציה f היא y = 2x + 3 ו- x = 5 , אז square(x) צריכה להחזיר 169 (13 בריבוע).

public Function derived() המחזירה את הנגזרת של f. למשל, אם f היא הפונקציה y = 2x + 3, על derived() להחזיר את הפונקציה הקבועה 2, ואם הפונקציה f היא הפונקציה הקבועה 2, על derived() להחזיר את הפונקציה הקבועה 0.

<u>סעיף ב' (5 נקודות)</u>

המחלקה LinearFunction המייצגת פונקציה ליניארית מהצורה y = ax + b.
המחלקה היא סוג של DerivedFunction.
המחלקה מגדירה בנאי המקבל כפרמטרים את a ו- b ממשיים.

<u>סעיף ג' (5 נקודות)</u>

המחלקה ConstantFunction המייצגת פונקציה קבועה מהצורה y = c.
המחלקה היא סוג של LinearFunction.
המחלקה מגדירה בנאי המקבל כפרמטר את c ממשי.

להלן תוכנית דוגמה (הפלט רשום כהערות).

```
public static void main(String[] args) {
        Function f1 = new LinearFunction(2, 3);
        System.out.println(f1);             // y = 2.0x + 3.0
        System.out.println(f1.valueAt(5));        // 13.0
        Function f2 = ((DerivedFunction) f1).derived();
        System.out.println(f2);             // y = 2.0
        System.out.println(f2.valueAt(5));        // 2.0
        Function f3 = ((DerivedFunction) f2).derived();
        System.out.println(f3);             // y = 0.0
        System.out.println(f3.valueAt(5));        // 0.0
}
```

שאלה 4 (20 נק')

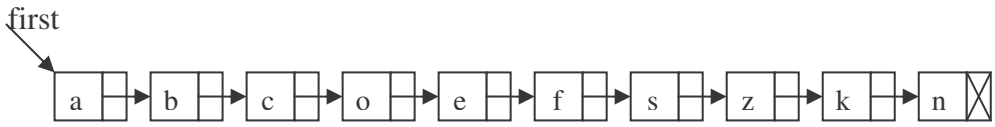
נתונות המחלקות Link ו-LinkedList הבאות:

```
public class Link {
        private Object data;
        private Link next;

        public Object getData() { return data;      }

        public void setData(Object o) { data = o; }

        public Link getNext() { return next; }

        public void setNext(Link next) { this.next = next; }
…
}

public class LinkedList {
        private Link first;
...
}
```

בכל אחד מסעיפים מותר להשתמש אך ורק בשיטות שצוינו בפירוש.

סעיף א' (5 נקודות)

כתבו במחלקה LinkedList שיטה בשם size אשר מחשבת ומחזירה את גודל הרשימה (מספר החוליות ברשימה).
לדוגמה, אם נפעיל את השיטה size על הרשימה:



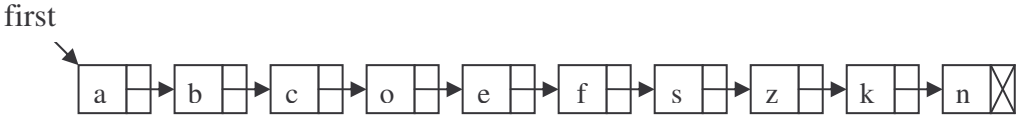
על השיטה להחזיר ערך 10.

```
public int size(){
        השלימו בדף התשובות//
}
```

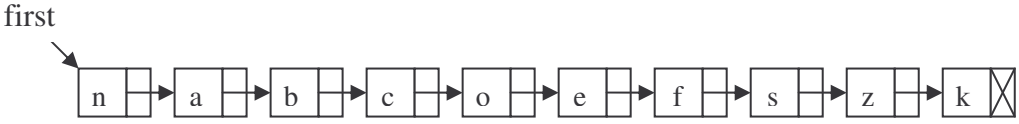
סעיף ב' (5 נקודות)

כתבו במחלקה LinkedList שיטה בשם moveLastToHead אשר מבצעת העברה של האיבר האחרון ברשימה לתחילת הרשימה (כלומר האיבר האחרון עובר להיות בראש הרשימה).

לדוגמה, אחרי שנפעיל שיטה moveLastToHead על הרשימה:



מתקבלת רשימה הבאה:



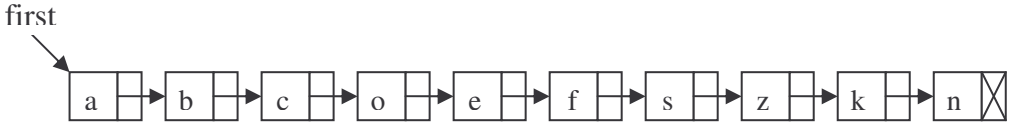

```
public void moveLastToHead(){
        השלימו בדף התשובות//
}
```

סעיף ג' (10 נקודות)

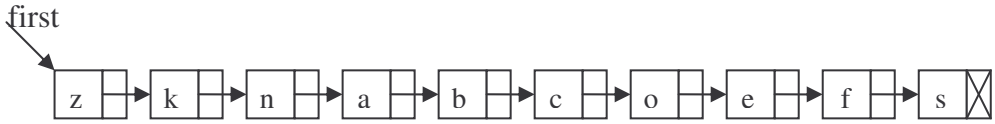
בסעיף זה ברצוננו להוסיף למחלקה LinkedList שיטה בשם rotateRight אשר מקבלת מספר שלם
אי שלילי k ומבצעת רוטציה של הרשימה k מקומות ימינה.
הערה: ניתן להניח ש-k>0 ו-k קטן מגודל הרשימה, ואין צורך לבדוק תנאי זה בתוך השיטה

שימו לב: ישנו פתרון לא יעיל הקורא לשיטה moveLastToHead מסעיף ב'. פתרון זה אינו יעיל ויזכה בניקוד חלקי בלבד.

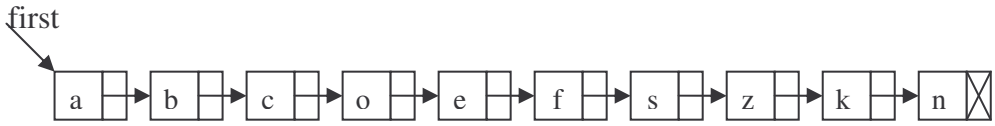
דוגמה 1:
אחרי שנפעיל את השיטה rotateRight(3) על הרשימה:



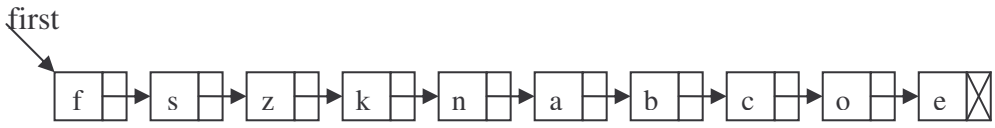
מתקבלת הרשימה הבאה:



דוגמה 2:
אחרי שנפעיל את השיטה rotateRight(5) על הרשימה:



מתקבלת הרשימה הבאה:




```
public void rotateRight (int k){
        השלימו בדף התשובות//
}
```

שאלה 5 (20 נקודות)

השלימו את השיטה subsetsOfSize(int n,int k), אשר מקבלת שני מספרים שלמים k ו- n (כך ש- $0 \leq k$, $0 \leq n$) ומדפיסה את כל תתי-הקבוצות בגודל k של הקבוצה {1,2,...,n} (שהיא ריקה אם n = 0). הדפסת תת-קבוצה פירושה הדפסת כל איבריה **בסדר עולה** (מופרדים על ידי רווחים) כמחרוזת.

למשל, הפונקציה main הנתונה קוראת ל- subsetsOfSize(n,k) עם ארגומנטים n = 5 ו- k=3, והתכנית תדפיס את המחרוזות הבאות (אחת לשורה בסדר כלשהו בין הקבוצות):

```
1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5
```

אל תשכחו לטפל במקרה ש k>n.

```
public class Subsets{
   public static void subsetsOfSize(int n, int k){
            subsetsOfSize(n,k,"");
   }

   public  static void subsetsOfSize(int n, int k, String s){

   //השלימו בדף התשובות
   }

   public static void main(String[] args){
     subsetsOfSize(5,3);
   }
 }
```

שאלה 6 (10 נקודות)

השאלה מתייחסת לתרגיל בית 5.

כפי ששמתם לב, בריצת האלגוריתם האבולוציוני בתרגיל 5, ה-fitness לא בהכרח יורד בכל דור (כלומר משתפר לכיוון ה-fitness המושלם בעל ערך אפס). הסיבה: ייתכן והפרט הטוב ביותר בדור הנוכחי לא שורד ולכן לא עובר לדור הבא, והפרט הטוב ביותר בדור הבא הוא בעל ערך fitness גבוה יותר (כלומר פחות טוב). ישנו מנגנון הקרוי אליטיזם הדואג לכך שה-fitness של הפרט הטוב בדור תמיד יהיה קטן או שווה ל-fitness של הפרט הטוב ביותר בדור שלפניו. בצורה זו, ערכי ה-fitness אינם יכולים לעלות מדור לדור — אלא רק לרדת (או לכל הפחות להישאר באותו ערך).

עקרון האליטיזם פשוט: העבר את הפרט הטוב ביותר בדור הנוכחי, כמות שהוא, לדור הבא.

עליכם לעדכן את הקוד שכתבתם בתרגיל 5 על מנת שיתמוך באליטיזם.

ענו בדך התשובות עך הסעיפים הבאים:

סעיף א' (3 נקודות)

האם חייבים במחלקה חדשה למימוש מנגנון האליטיזם או שהמנגנון ימומש בתוך מחלקה קיימת? (ענה "מחלקה חדשה" או נקוב בשמה של המחלקה הקיימת) ?

סעיף ב' (3 נקודות)

האם חייבים בשיטה חדשה למימוש מנגנון האליטיזם או שהמנגנון ימומש בתוך שיטה קיימת? (ענה "שיטה חדשה" או נקוב בשמה של השיטה הקיימת) ?

סעיף ג' (4 נקודות)

תאר במשפט אחד קצר כיצד ימומש מנגנון האליטיזם.

להלן **חלק** מההגדרות של תרגיל 5 לשימושכם.

```
public class RegressionEvolution extends Evolution{…}
public class Evolution {
        public Individual getBest()…
        public void evolve()…
}
public abstract class Individual implements Comparable, Replicable, Variable{
        public double getFitness()…
        protected abstract double evaluate()…
        public boolean isIdeal()…
        public int compareTo(Object obj)…
        protected abstract Individual copy()…
        protected abstract void initGenome()…
        public Object replicate()…
}
public class RegressionIndividual extends Individual{…}
public interface Selection {
        Individual reproduce(Individual[] pop);
}
public interface Replicable {
        public Object replicate();
}
public class Population {
        public void nextGeneration()…
        public Individual getBest()…
}
public interface Variable {
          Individual mutate();
          Individual crossover(Individual other);
}
public class RankSelection implements Selection{…}
```