תאריך הבחינה: 16.1.2011

שמות המרצים: מר שי זקוב
ד"ר פז כרמי
פרופ' מייק קודיש
ד"ר חן קיסר
ד"ר צחי רוזן

שם הקורס: מבוא למדעי המחשב

מספר הקורס: 202-1-1011

שנה: 2011 סמסטר: א' מועד: א'

משך הבחינה: **3 שעות**

חומר עזר: **אסור**

מבוא למדעי המחשב

202-1-1011

מבחן

אנא קראו את ההוראות שלהלן בעיון:

- במבחן זה 7 שאלות. ענו על כולן.
- בשאלות **שבהן לא מצוין אחרת**, הנכם יכולים לבחור בפתרון רקורסיבי או בפתרון לא רקורסיבי.
- רשמו את תשובותיכם **בדפי התשובות בלבד**. המחברת שקיבלתם היא מחברת טיוטה, והיא לא תימסר כלל לבדיקה. בסיום הבחינה נשמור אך ורק את דפי התשובות. כל שאר החומר יועבר לגריסה.
- החשוב ביותר הוא נכונות הקוד. עם זאת, יעילות, סגנון, וכתיבה ברורה חשובים גם הם, ולכן **תשובה לא יעילה או לא ברורה** עלולה לגרום להפחתת נקודות.
- בשאלות התכנות המקום בדף התשובות **מספיק** עבור תשובה המנוסחת כראוי.
- הקפידו על כתב יד ברור. **תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא**. כיתבו פתרון לשאלה קודם כול במחברת הטיוטה, ורק לאחר מכן העתיקו פתרון נקי לדף התשובות.
- **אין להוסיף פונקציות עזר** אלא אם כן הדבר נתבקש באופן מפורש בשאלה.
- אין לרשום הערות אך יש לדאוג שהקוד יהיה ברור לחלוטין ללא הערות (ריווח, שמות משתנים, וכד').
- אנא קראו שאלה עד תומה בטרם תענו עליה.
- **הקפידו לרשום בכל דפי התשובות גם את מספר הנבחן ואת מספר החדר שבו אתם נבחנים.**
- אם אינכם יודעים את התשובה לשאלה כלשהי, רשמו "לא יודע/ת" (במקום תשובה) ותזכו ב- 20% מניקוד השאלה. **אין אפשרות לרשום "לא יודע" על סעיף — רק על שאלה שלמה.**

בהצלחה!

<u>שאלה 1 (14 נקודות)</u>

פגשנו את מושג הסדר המילוני (או "הלקסיקוגרפי") במהלך עבודת בית מספר 3. כתבו פונקציה lexLT בעלת התבנית הבאה אשר מחזירה true אם מחרוזת s1 קודמת למחרוזת s2 בסדר המילוני, false אחרת.

```
public static boolean lexLT(String s1, String s2){
      boolean answer;
      // הקוד שלכם כאן
      return answer;
}
```

לדוגמא:

המחרוזת "abc" קודמת למחרוזת "quarsbecy" אך לא למחרוזת "aabc".
המחרוזת "abbc" קודמת למחרוזת "abbec" אך לא למחרוזת "abbac".
המחרוזת "ab" קודמת למחרוזת "abbec".

שום מחרוזת לא קודמת לעצמה. המחרוזת הריקה קודמת לכל מחרוזת שאינה ריקה.
ניתן להניח כי המחרוזות s1 ו- s2 אינן null ומכילות אותיות קטנות באנגלית בלבד .
השלימו את גוף הפונקציה lexLT בדף התשובות.
<u>אסור להשתמש</u> ב-compareTo או ב-compare.

<u>שאלה 2 (11 נקודות)</u>

נתון הקוד הבא:

```
public static void sortSederYored(int[] arr)
{
      if (arr != null && arr.length>1){
            int n = arr.length;
            int i = n-2;
            while(i>=0){
                  insert(arr, i);
                  i = i-1;
            }
      }
}
public static void insert(int[] arr, int i) {
      int value = arr[i];
      while (i<arr.length-1 && value<arr[i+1]) {
            arr[i] = arr[i+1];
            i = i+1;
      }
      arr[i] = value;
}
```

ציינו עבור כל אחת מהטענות המופיעות בדף התשובות האם היא טענה נשמרת או לא של לולאת ה-while שבפונקצייה sortSederYored. (הערה: [a,b] מסמן את הערכים שבין a ל-b, <u>כולל</u> a <u>וכולל</u> b.)

<u>שימו לב</u>: בשאלה זו הניקוד עבור סעיף הוא נקודה אחת עבור תשובה נכונה, 0 נקודות עבור אי-סימון תשובה, <u>ותורד חצי נקודה</u> עבור תשובה שגויה. הציון המינימאלי הוא 0 (כלומר לא יינתן בכל מקרה ציון שלילי).

שאלה 3 (20 נקודות)

בהינתן מערך (לא ריק) $\{x_0,x_1,\ldots,x_{n-1}\}$ `int[] x =` וערך `value` מסוג `int` יש להחליט האם קיימים מקדמים $a_i\in\{-1,0,1\}$ עבור $0\le i\le n-1$ כך שערכו של הביטוי החשבוני $a_0x_0 + a_1x_1 + \ldots + a_{n-1}x_{n-1}$ הינו `value`.

<u>דוגמאות:</u>
אם `x ={2,3,6,7,10}` ו- `value=(-5)`, קיימים מקדמים שכן
`1*2 + (-1)*3 + 1*6 + 0*7 + (-1)*10 = -5`.

אם `x ={5,14,7,3}` ו- `value=20`, לא קיימים מקדמים כנדרש.

הפונקציה `rec` מחזירה `true` במידה וקיימים מקדמים כנדרש, `false` אחרת:

```
public static boolean rec(int[] x, int value){
      return recSolve (0 , x, value);
}
public static boolean recSolve(int i, int[] x, int value){
      boolean answer;
      // הקוד שלכם כאן
      return answer;

}
```

השלימו את הפונקציה הרקורסיבית `recSolve` בדף התשובות.
ניתן להניח כי המערך `x` אינו `null`.

שאלה 4 (13 נקודות)

הממשק Filter מאפשר לנו לסנן איברים.

```
public interface Filter{
      public boolean accept (Object obj);
}
```

קריאה לשיטה accept(obj) תחזיר ערך בוליאני המסמן האם "לקבל" את העצם obj (אם הערך המוחזר הוא true) או לא.

בעבודה 4 כתבתם את הממשק Expression וכן מספר מחלקות המממשות את הממשק.

תזכורת לממשק Expression:

חתימה	תכונה
String getLastOperationSymbol()	החזרת מחרוזת המתארת את סמל הפעולה האחרונה בביטוי
double evaluate()	שיערוך והחזרת ערכו של הביטוי
Expression derivative(Variable var)	החזרת הנגזרת של הביטוי המתמטי לפי המשתנה var

כתבו את המחלקה FilterValueExpression המממשת את הממשק Filter.

על השיטה accept במחלקה לקבל רק עצמים מטיפוס Expression שערכם גדול או שווה ל-0.

שאלה 5 (12 נקודות)

תזכורת: הממשק Comparator מגדיר שיטה יחידה, אשר חתימתה היא:

```
public int compare(Object o1, Object o2);
```

אוביקטים מטיפוס Comparator מגדירים סדר על קבוצה של אוביקטים מסוג מסוים. הפעלת השיטה compare על הארגומנטים o1 ו-o2 תחזיר מספר שלילי אם o1 קודם בסידור, מספר חיובי אם o2 קודם בסידור, ו-0 אם o1 ו-o2 שקולים מבחינת הסידור. אם טיפוס אחד האוביקטים o1 או o2 אינו מתאים לקבוצה עליה הסדר מוגדר, השיטה זורקת חריגה.

התבוננו בקוד הבא:

```
public static void main(String[] args) {
      Integer[] integers = {
            new Integer(4), new Integer(3), new Integer(5), new Integer(1),
            new Integer(2)
      };

      Lecturer[] lecturers = {
            new Lecturer("Moshe"), new Lecturer("Paz"), new Lecturer("Mike"),
            new Lecturer("Chen"), new Lecturer("Tzachi"), new Lecturer("Shay")
      };

      Comparator intComp = new IntegerComparator();
      Comparator lecComp = new LecturerComparator();

      sort(integers, intComp);
      sort(lecturers, lecComp);
      printArray(integers);  // {1,2,3,4,5}
      printArray(lecturers); // {Chen,Mike,Moshe,Paz,Shay,Tzachi}

      Comparator revIntComp = new ReverseComparator(intComp);
      Comparator revLecComp = new ReverseComparator(lecComp);

      sort(integers, revIntComp);
      sort(lecturers, revLecComp);
      printArray(integers);  // {5,4,3,2,1}
      printArray(lecturers); // {Tzachi,Shay,Paz,Moshe,Mike,Chen}
}
```

בקוד לעיל, המחלקות IntegerComparator, LecturerComparator, ו-ReverseComparator ממשות את הממשק Comparator. המחלקה IntegerComparator מגדירה סדר על אוביקטים מטיפוס Integer, כאשר מספר בעל ערך נמוך קודם בסידור למספר בעל ערך גבוה. המחלקה LecturerComparator מגדירה סדר על אוביקטים מטיפוס Lecturer, כאשר מרצים מסודרים לפי הסדר המילוני (כמו בשאלה 1) של שמם. השיטה הסטטית sort נתונה לכם: היא מקבלת מערך של אוביקטים ואוביקט מטיפוס Comparator, ומסדרת את האוביקטים במערך לפי הסידור שה-Comparator מגדיר. השיטה הסטטית printArray נתונה לכם: היא מקבלת מערך של אוביקטים, ומדפיסה את איבריו לפי סדרם במערך. ליד כל קריאה ל-printArray בקוד, מופיעה הערה המציגה את הפלט שהתקבל מקריאה זו.

השלימו את הקוד החסר במחלקה ReverseComparator שבדף התשובות.

<u>אין לכתוב את המילים Integer ו-Lecturer בקוד המחלקה</u> (שימוש במילים אלו יגרום ל<u>פסילת</u> התשובה).

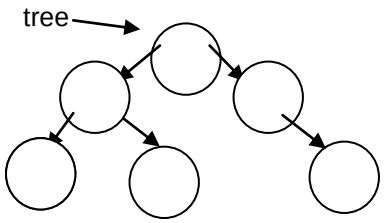
שאלה 6 (15 נקודות)

שאלה זו עוסקת בעצים בינאריים כפי שנלמדו בכיתה. נתון:

```
public class  BinaryTree {
      private  BinaryNode root;

      public int countNodesAtDepth (int depth) {
            int ans = 0;
            if (root != null)
                  ans = root.countNodesAtDepth(depth);
            return ans;
      }
      //שאר המחלקה כפי שראינו בכיתה
}

public  class  BinaryNode {
      protected   Object data;
      protected   BinaryNode left;
      protected   BinaryNode right;

      //השלם שיטה זו ע"פ הנדרש בשאלה
      public int countNodesAtDepth (int depth) {
            // הקוד שלך כאן
      }

      //שאר המחלקה כפי שראינו בכיתה
}
```

נתונה השיטה לעיל `countNodesAtDepth` במחלקה `BinaryTree`.
עליכם להשלים את השיטה `countNodesAtDepth` במחלקה `BinaryNode`.
השיטה מקבלת פרמטר depth מטיפוס int, ומחזירה את מספר הצמתים בעומק depth יחסית לצומת עליו מופעלת השיטה. אם אין לצומת צאצאים ברמת העומק היחסי הנתונה, על השיטה להחזיר 0.
ניתן להניח כי `depth` הוא מספר אי שלילי.

לדוגמא, נניח כי המשתנה `tree` הוא מטיפוס `BinaryTree` והוא מצביע לעץ הבא:



הפעלת השיטה tree.countNodesAtDepth (0) תחזיר 1.
הפעלת השיטה tree.countNodesAtDepth (1) תחזיר 2.
הפעלת השיטה tree.countNodesAtDepth (2) תחזיר 3.
הפעלת השיטה tree.countNodesAtDepth (3) תחזיר 0.

שאלה 7 (15 נקודות)

יוסי רוצה לטפס על סולם.
בכל צעד כלפי מעלה הוא יכול לטפס שלב אחד או שניים.
בכמה אופנים שונים יכול יוסי לטפס על סולם בעל n שלבים, כאשר הוא מתחיל מהרצפה ?
ניתן להניח כי בסולם שלב אחד לפחות.

לדוגמה:
מספר האופנים השונים לטפס על סולם בעל שני שלבים הוא: 2.
(רצפה->שלב 1->שלב 2 וכן רצפה->שלב 2)
מספר האופנים השונים לטפס על סולם בעל שלשה שלבים הוא: 3.
(רצפה->שלב 1->שלב 2->שלב 3, רצפה->שלב 1->שלב 3, רצפה->שלב 2->שלב 3)

עליך לכתוב פונקציה לחישוב מספר האופנים בהנתן n.