מבוא למדעי
המחשב
202-1-1011

מבוא לתכנות
למערכות מידע
202-1-1041

סמסטר א' תשס''ט
מבחן מועד ב'

מר איתי בר-יוסף
מר טל גרינשפון
גב' דיקלה דותן
ד''ר מיכל זיו יוקלסון
פרופ' משה זיפר
מר שי זקוב
פרופ' מיכאל קודיש
ד''ר צחי רוזן

משך המבחן: שלוש וחצי שעות.
חומר עזר אסור.

במבחן זה 5 שאלות. יש לענות על כולן.

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. מותר להסתמך על סעיפים קודמים גם אם לא עניתם עליהם.

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.

בדף האחרון של המבחן ישנה תזכורת למספר פונקציות חשובות, בהן תוכלו להשתמש במהלך המבחן.

בהצלחה!!!

שאלה 1 (20 נקודות)

השלימו את הפונקציה numbers(int n, int b) אשר מקבלת שני מספרים שלמים n ו- b ומדפיסה את כל המספרים בעלי n ספרות בבסיס b. הדפסת מספר פירושה הדפסת כל ספרותיו כמחרוזת (כולל אפסים על מנת להשלים ל- n ספרות). יש להניח ש- $0 \leq n$ ו- $1 \leq b < 10$.

<u>שימו לב:</u> לא ניתן להוסף פונקציות נוספות.

למשל, הפונקציה main הנתונה קוראת ל-numbers(int n, int b) כאשר n=2 ו- b=3 , ומדפיסה את המחרוזות הבאות (אחת לשורה בסדר כלשהו):

21
22
20
12
11
10
02
01
00

המייצגות את כל המספרים בעלי 2 ספרות בבסיס 3.

עליכם להשלים את הארגומנטים שמקבלת הפונקציה הרקורסיבית numbersRec (סעיף ב'), את הערכים הנשלחים אליה לראשונה (סעיף א') וכן את הפונקציה הרקורסיבית עצמה(סעיף ג').

```
public class numbers{
	public static void numbers(int n, int b){
		numbersRec (/* השלם בדף התשובות - (2 נקודות) סעיף א */);
	}
	public static void numbersRec(/* השלם בדף התשובות - (2 נקודות) סעיף ב */){

		/* השלם בדף התשובות - (16 נקודות) סעיף ג */

	}
	public static void main(String[] args){
		numbers(2,3);
	}
}
```

שאלה 2 (25 נק')

שאלה זו עוסקת בעצים בינאריים בהם השדה Object data הינו מסוג Integer. (תוכלו למצא תזכורת לגבי המחלקה Integer בדף המידע שבסוף המבחן.)
נתונות לכם המחלקות BinaryNode ו- BinaryTree, כפי שנלמדו בכיתה .

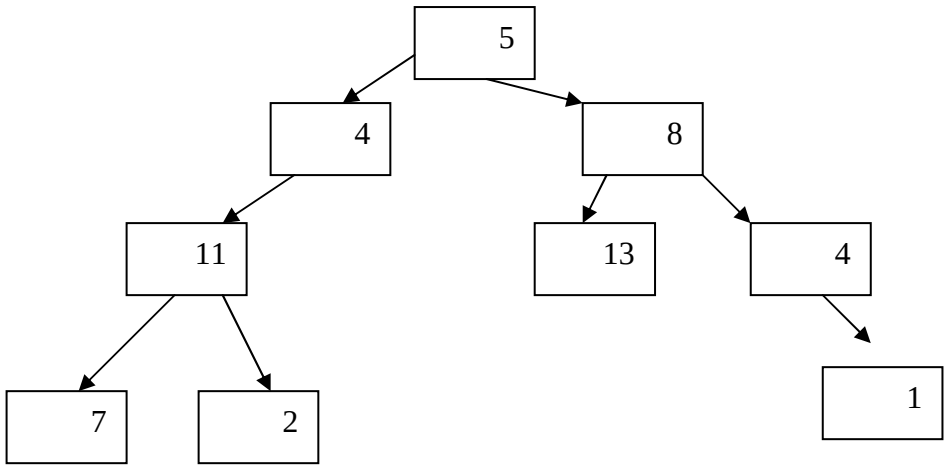
```
public class BinaryNode {
   public Object data;
   public BinaryNode left;
   public BinaryNode right;

   public BinaryNode (Object data) {
      this.data = data;
      left = null;
      right = null;
    }
}
public class BinaryTree{
    private BinaryNode root;

    public BinaryTree(){
        root=null;
    }
}
```

הגדרה: מסלול **שורש-עלה** הוא מסלול המחבר בין שורש העץ לאחד מן העלים בעץ.

לדוגמא, בעץ הבא ישנם ארבעה מסלולי **שורש-עלה**:



מסלולי שורש העלה הם:

1. 5→4→11→7
2. 5→4→11→2
3. 5→8→13
4. 5→8→4→1

<u>סעיף א' (10 נקודות)</u>
בסעיף זה נעסוק בסכום ערכי הקודקודים לאורך מסלול שורש-עלה.

עליכם להוסיף את השיטה boolean hasPathSum(int sum) למחלקה BinaryNode, אשר בהינתן עץ בינארי ומספר שלם כלשהוא, מחזירה ערך **true** אם ורק אם קיים מסלול שורש עלה אשר סכום ערכי קודקודיו שווה לערך השאילתה.

למשל עבור העץ בדוגמא והערך 27, השיטה תחזיר ערך true, מכיוון שסכום ערכי הקודקודים במסלול (1) שווה ל-27 (27=5+4+11+7).

אנו נוסיף למחלקה BinaryTree את השיטה הבאה:

```
public class BinaryTree{
        .
        .
        public boolean hasPathSum(int sum){
                return root.hasPathSum(sum);
        }
}
```

עליכם להוסיף (בדף התשובות) למחלקה BinaryNode את השיטה הבאה:

```
class BinaryNode {
        .
        .
        public boolean hasPathSum(int sum){

                /*סעיף א (10 נקודות) -השלם בדף התשובות */

        }
}
```

<u>שימו לב:</u> לא ניתן להוסיף פונקציות נוספות.

המשך השאלה בעמוד הבא

<u>סעיף ב' (15 נקודות)</u>

עליכם לתכנן שיטה שתאפשר לעץ בינארי להדפיס את כל מסלולי שורש-העלה שבו.
לדוגמא, עבור העץ בדוגמה, יתקבל הפלט:

```
5 4 11 7
5 4 11 2
5 8 13
5 8 4 1
```

עליכם להוסיף את השיטה printPaths למחלקה BinaryTree(**סעיף ב1**). שיטה זו נעזרת בשיטה נוספת printPaths שתוסיפו למחלקה BinaryNode(**סעיף ב2**). בשאלה זו **לא ניתן** להשתמש בשיטות עזר (שני הסעיפים).

```
class BinaryTree{
    .
    .
    public void printPaths(){

        root.printPaths(/*סעיף ב1 (3 נקודות) -השלם בדף התשובות */);
    }
}
```

במחלקה BinaryNode :

```
class BinaryNode{
    .
    .
    public void printPaths(/*סעיף ב2 (3 נקודות) -השלם בדף התשובות */){

                    /*סעיף ב (9 נקודות) -השלם בדף התשובות */

    }
}
```

שאלה 3 (20 נק')

בשאלה זו נעסוק בפירוק של מחרוזות ליחידות קטנות (Tokens) באמצעות מנגנון ה-Iterator :

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

עליכם לממש את המחלקות הבאות:

סעיף א (13 נקודות)

ממשו את המחלקה StringTokenIterator אשר בהינתן מחרוזת של אותיות (משפט), בכל קריאה לשיטה next, נקבל את המילה הבאה (תת-המחרוזת הבאה המופרדת ע"י רווח).

שימו לב: ניתן להניח כי בקלט המתקבל, כל שתי מלים מופרדות ע"י רווח אחד בדיוק, ולא תינתן כקלט מחרוזת ריקה או מחרוזת המורכבת מרווחים בלבד.

לדוגמא:

```
Iterator iter=new StringTokenIterator ("Hello my name is shlomo");

while(iter.hasNext())
        System.out.print(iter.next()+"*");
```

הפלט המתקבל מקטע קוד זה:

```
Hello*my*name*is*Shlomo*
```

השלימו את המחלקה:

```
class StringTokenIterator implements Iterator{

                /* הוסיפו משתני מחלקה, סעיף א1 (1 נקודות) -השלם בדף התשובות */
        /* constructor */
         public  StringTokenIterator(  /* סעיף א2 (1 נקודות) -השלם בדף התשובות */ ){

                /*סעיף א3 (2 נקודות) -השלם בדף התשובות */
        }

        /* class methods */
         public boolean hasNext(){

                /*סעיף א4 (2 נקודות) -השלם בדף התשובות */

        }

        public Object next(){

                /*סעיף א5 (7 נקודות) -השלם בדף התשובות */

        }
}
```

המשך השאלה בדף הבא

<u>סעיף ב (7 נקודות)</u>

ממשו את המחלקה StringTokenReverseIterator אשר ממשת את מנגנון ה- Iterator באופן דומה, אך מחזירה את המילים בסדר הפוך (מהסוף להתחלה). בשאלה זו **לא ניתן** להשתמש בשיטות עזר (שני הסעיפים).

המחלקה StringTokenReverseIterator נעזרת במחלקה StringTokenIterator מסעיף א, ומשתמשת ב- interface של מחסנית (Stack), הממומש ע"י המחלקה StackAsArray, לפתרון הבעיה. **<u>שימו לב:</u>** ניתן להשתמש בסעיף א׳ גם אם לא פתרתם אותו.
תזכורת לגבי ממשק המחסנית תוכלו למצא בדף האחרון.

השלימו את המחלקה:

```
class StringTokenReverseIterator  implements Iterator{

        private Stack st;



    public StringTokenReverseIterator ( /* השלם בדף התשובות -(1 נקודות) 1ב סעיף */ ){

                st = new StackAsArray();

                Iterator iter=  /* בשורה זו ניצור משתנה בשם iter  ובהמשך נשתמש בו */

                 /* סעיף ב2 ( 2 נקודות)-השלם בדף התשובות */

        }
        public boolean hasNext(){

                 /* סעיף  ב3 (2 נקודות)-השלם בדף התשובות */

        }

        public Object next(){

                 /* סעיף ב4  (2 נקודות)-השלם בדף התשובות */

        }
}
```

לדוגמא:

```
Iterator  iter=new StringTokenReverseIterator ("Hello my name is shlomo");

while(iter.hasNext())
        System.out.print(iter.next()+"*");
```

עתה, הפלט המתקבל מהקוד הבא:

```
Shlomo*is*name*my*Hello*
```

שאלה 4 (20 נק')

שאלה זו עוסקת במיון איברי מערך. בכל סעיפי השאלה ניתן להניח כי מערך הקלט אינו ריק ואינו null וכי המערך מכיל מספרים שלמים אי-שליליים. כמו כן ניתן להניח שכל שאר הקלט חוקי, כמוסבר בסעיפים השונים (למשל, בסעיף ג' אין צורך לבדוק אם i> 0) .

סעיף א' (5 נקודות)

ממשו את הפונקציה

public static int maxNumDigits (int[] arr)

המחזירה את מספר הספרות המקסימאלי של מספר במערך arr.

לדוגמא,

עבור המערך | 5 | 5 | 0 | 7 | יוחזר הערך 1 (שכן גם במספר הגדול ביותר ישנה רק ספרה אחת),

ועבור המערך | 51 | 5 | 520 | 7 | יוחזר הערך 3 שכן המספר הגדול ביותר הוא בעל 3 ספרות.

סעיף ב' (5 נקודות)

יהיו x ו-y שני מספרים שלמים אי-שליליים. נאמר ש- x גדול מ-y בספרה ה-i, אם ערך הספרה ה- i במספר x גדול מערך הספרה ה- i במספר y, כאשר הספרה ה-1 מייצג את ספרת האחדות, הספרה ה-2 מייצג את ספרת העשרות, וכך הלאה. במקרה ולמספר יש פחות מ-i ספרות נאמר שערך הספרה ה- i של המספר הוא 0.

לדוגמא, עבור המספרים x=4031, y=22, y גדול מ-x בספרה ה-1 ואילו x גדול מ-y בספרה ה-2 ובספרה ה-4. אף אחד משני המספרים לא גדול מהשני בספרה ה-3.

ממשו את הפונקציה

public static boolean greaterAtPosition (int x, int y, int i)

המקבלת שני מספרים אי-שליליים x ו-y ומספר i>0 ומחזירה ערך true אם ורק אם המספר x גדול מהמספר y בספרה ה-i.

סעיף ג' (5 נקודות)

ממשו את הפונקציה

public static void sort(int [] arr, int i)

המקבלת מערך arr ומספר i>0, וממיינת את איברי המערך arr לפי ערכי הספרה ה-i .

לדוגמא,

עבור המערך | 82 | 7 | 52 | 6 | תוצאת המיון עבור i=1 תהיה: | 82 | 52 | 6 | 7 |

בסעיף זה ניתן להתבסס על כל אחת משיטות המיון שנלמדו בכיתה. כמו כן, ניתן להוסיף פונקציות עזר.

<u>סעיף ד' (5 נקודות)</u>

מיון לפי ספרות היא שיטת מיון בה ממיינים את המערך לפי ספרות, החל מהספרה ה-1 (ספרת האחדות), הספרה ה-2 (ספרת העשרות) וכך הלאה.

לדוגמא,

עבור המערך

51	17	520	15

בשלב הראשון נמיין את המספרים לפי הספרה ה-1 (ספרת האחדות) ויתקבל המערך הבא:

520	51	15	17

בשלב השני ימוין המערך לפי הספרה ה-2 (ספרת העשרות) ויתקבל המערך הבא:

15	17	520	51

בשלב השלישי (והאחרון בדוגמא זו) ימוין המערך לפי הספרה ה-3 (ספרת המאות) ויתקבל המערך הבא:

15	17	51	520

המהווה מערך ממוין.

ממשו את הפונקציה

```
public static void sortByPosition(int [] arr)
```

המקבלת מערך arr וממיינת אותו בשיטת מיון לפי ספרות. ניתן להשתמש בסעיפים הקודמים גם אם לא פתרתם אותם.

שאלה 5 (15 נקודות)

בשאלה זו 4 סעיפים (א-ד), אותם יש למלא בדף התשובות. בדף התשובות תמצאו סעיפים המתיחסים לקוד הבא, הכולל את המחלקות Coconut, Swallow, ו-EuropeanSwallow, וכן המחלקה Run הכוללת main:

```
public class Coconut {
	public int weight;

	public Coconut(int weight) {
		this.weight = weight;
	}
} //class Coconut

public abstract class Swallow {
	public int speed;
	public Coconut coconut;

	public Swallow(int speed, Coconut coconut) {
		this.speed = speed;
		this.coconut = coconut;
	}
	public Swallow(Swallow toCopy){
		speed = toCopy.speed;
		this.coconut = toCopy.coconut;
	}
	public String getInfo(){
		return "Swallow speed: " + speed +
			", coconut weight: " + coconut.weight;
	}
	public abstract String getType();
	private String greeting1() {return "Hello.";}
	public String greeting2() {return "How are you?";}
	public String toString(){
		return greeting1() + " I'm a " + getType() +
			" swallow. " + greeting2();
	}
} //class Swallow

public class EuropeanSwallow extends Swallow {
	public EuropeanSwallow(int speed, Coconut coconut) {
		super(speed, coconut);
	}
	public EuropeanSwallow(Swallow toCopy) {
		super(toCopy);
	}
	public String getType() {return "European";}
	public String greeting1() {return "Good morning. ";}
	public String greeting2() {return "How do you do?";}
} //class EuropeanSwallow
```

```
public class Run {
	public static void main(String[] args) {
		Coconut coco = new Coconut(100);
		EuropeanSwallow twity =
			new EuropeanSwallow(15, coco);
		EuropeanSwallow shraga =
			new EuropeanSwallow(twity);
		twity.speed = 20;
		twity.coconut.weight = 200;
		<a>
	}
} //class Run
```

בהצלחה

הגדרה חלקית של המחלקה Integer:

```
public class Integer{
        private int value;
        public Integer( int value ){this.value =value; }
        public int intValue(){ return value; }
        public String toString(){ return value + ""; }
}
```

הממשק St ack :

```
public interface Stack {
    public void push(Object o);
     public Object pop();
    public boolean isEmpty();
}
```

שיטות במחלקה String:

• int **length**()

שיטה המחזירה את אורך המחרוזת.

• int **indexOf**(char c)

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

• char **charAt**(int index)

שיטה המחזירה את התו במיקום index במחרוזת.

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

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

שיטות המחלקה Math:

• static double **pow**(double a, double b)

פונקציה המחזירה את הערך a בחזקת b.

• static double **sqrt**(double a)

פונקציה המחזירה את השורש הריבועי החיובי של a.

• static double **abs**(double a)

פונקציה המחזירה את ערכו המוחלט של a.

• static int **abs**(int a)

פונקציה המחזירה את ערכו המוחלט של a.

• static double **min**(double a, double b)

פונקציה המחזירה את הערך המינימאלי מבין a ו-b.

• static int **min**(int a, int b)

פונקציה המחזירה את הערך המינימאלי מבין a ו-b.

• static double **max**(double a, double b)

פונקציה המחזירה את הערך המקסימאלי מבין a ו-b.

• static int **max**(int a, int b)

פונקציה המחזירה את הערך המקסימאלי מבין a ו-b.

• static double **random**()

פונקציה המחזירה מספר רנדומאלי $0 \le r < 1$,r