מבוא למדעי המחשב
202-1-1011

מבוא לתכנות למערכות מידע
202-1-1041

סמסטר א' תשס''ט
מבחן מועד א'

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 הרקורסיבית subsetsOfSize(int n, int k) אשר מקבלת שני מספרים שלמים n ו-k, ומדפיסה את כל תתי הקבוצה בגודל k של הקבוצה {1,2,. . .,n} (הדפסת תת-קבוצה פירושה הדפסת כל איבריה כמחרוזת, בסדר כלשהו). ניתן להניח כי $0 \le k \le n$ ו- $1 \le n < 10$. לא ניתן להוסיף פונקציות נוספות.

למשל, הפונקציה main הנתונה קוראת ל-subsetsOfSize(int n, int k), כאשר n=5 ו-k=3, ומדפיסה את המחרוזות הבאות (בסדר כלשהו):

```
123
124
125
134
135
145
234
235
245
345
```

```
public class Subsets{
	public static void subsetsOfSize (int n, int k){
		subsetsOfSizeRec (/* השלם בדף התשובות - (3 נקודות) סעיף א*/);
	}

	public static void subsetsOfSizeRec(/*סעיף ב (3 נקודות)- השלם בדף התשובות*/){
		/* השלם בדף התשובות - (14 נקודות) סעיף ג*/
	}

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

שאלה 2 (20 נק')

נתונה המחלקה BinaryNode הבאה:

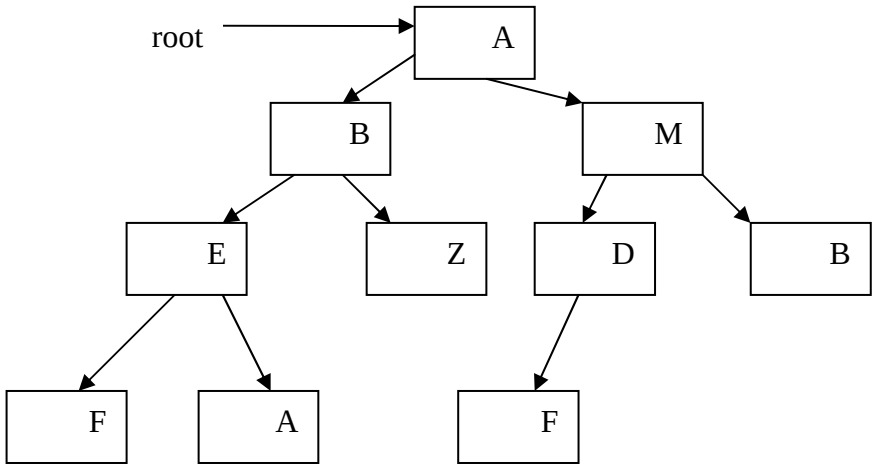
```
public class BinaryNode {
        public Object data;
        public BinaryNode left;
        public BinaryNode right;

        public BinaryNode (Object data) {
                data = data;
                left = null;
                right = null;
        }
}//class
```

בשאלה זו יש לממש את השיטה **הרקורסיבית** getPath במחלקה BinaryNode, אשר מקבלת אובייקט (obj), **ומחזירה מחרוזת** המייצגת את המסלול בעץ מהקודקוד הנוכחי (עצם המפתח, this) אל הקודקוד שמכיל אובייקט הזהה ל-obj. אובייקט data זהה ל-obj אם data.equals(obj) מחזירה true.

במידה והאובייקט לא נמצא, השיטה מחזירה מחרוזת ריקה, אחרת מחזירה מחרוזת המתקבלת ע"י שרשור ערכי ה-()toString של האובייקטים הנמצאים בקודקודים במסלול, מופרדים ע"י רווחים. ראו דוגמה.

ניתן להניח כי בקודקודי העץ לא נמצאת data שהיא null, וכי הארגומנט המתקבל (obj) איננו null. שימו לב כי אובייקט יכול להופיע מספר פעמים בעץ. במקרה זה, השיטה תחזיר אחד מהמסלולים האפשריים (אין חשיבות איזה מבין המסלולים).

לדוגמה, עבור העץ הבא, אשר המשתנה BinaryNode root מהווה רפרנס אליו (מצביע לשורש העץ):



מסלולים אפשריים המוחזרים עבור מספר קלטים (ייתכן שיחזרו מסלולים אחרים) :

root.getPath("F")→ "A B E F "
root.getPath("A")→ "A "
root.getPath("B")→ "A B "

ממשו את השיטה הבאה במחלקה BinaryNode:

```
public String getPath(Object obj) {
        /* השלם בדף התשובות - (20 נקודות)*/
}
```

שאלה 3 (20 נק')

שאלה זו עוסקת במערך של אובייקטים, ובממשק Iterator הנתון:

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

עליכם להשלים את המחלקה RandomIterator, המגדירה איטרטור עבור מערכים של אובייקטים, כלומר: Object[ ]. בכל קריאה לשיטה **next**, יוחזר אחד מאיברי המערך באופן **אקראי**, כאשר **אין** להחזיר אותו איבר פעמיים. כל עוד לא הוחזרו כל אברי המערך ע"י השיטה next, על השיטה **hasNext** להחזיר true. לאחר שהאיטרטור עבר על כל אברי המערך, על השיטה **hasNext** להחזיר false. יש להשתמש בפונקציה Math.random() אשר מחזירה מספר אקראי r, $0 \le r < 1$.

לדוגמה עבור הקוד הבא:

```
Sring[] arr = {"A","B","C","D" };
Iterator iter = new RandomIterator(arr);
while(iter.hasNext())  System.out.print(iter.next().toString() + " ");
```

אחד הפלטים האפשריים הוא:

B D A C

בהגדרת המחלקה RandomIterator, הנתונה לכם, מוגדר מערך בשם **arr**. מערך זה מאותחל בבנאי של המחלקה (ראו בהמשך). שימו לב כי:

א. סדר החזרת האיברים ע"י השיטה next חייב להיות אקראי.

ב. אין לשנות את סדר אברי המערכים arr ו-inputArr (פרמטר לבנאי, ראו קוד בהמשך) **בתוך הבנאי (בסעיף ב)**,

ג. ניתן לשנות את סדר האיברים במערך arr בסעיפים האחרים.

```
public class RandomIterator implements Iterator{
        Object[] arr;
        /* סעיף א' (5 נקודות)  השלם בדף התשובות */

        public RandomIterator(Object[] inputArr){
                arr = new Object[inputArr.length];
                for (int i=0; i<inputArr.length; i++)
                        arr[i]=inputArr[i];

                /* סעיף ב' (5 נקודות)  השלם בדף התשובות */
        }

        public Object next(){
                /* סעיף ג' (5 נקודות)  השלם בדף התשובות */
        }

        public boolean hasNext(){
                /* סעיף ד' (5 נקודות)  השלם בדף התשובות */
        }
}//class RandomIterator
```

שאלה 4 (20 נק')

שאלה זו עוסקת ברשימה ממוינת (SortedList) עם sentinel. מבנה הרשימה דומה לרשימה רגילה, אך ההבדל העיקרי הוא ששדה data הוא מטיפוס Comparable. האיברים ברשימה שמורים בצורה ממוינת, כאשר האיבר הקטן ביותר נמצא בראש הרשימה. נתונה לכם המחלקה ComparableLink הבאה:

```
public class ComparableLink {
        public ComparableLink next;
        public Comparable data;

        public ComparableLink (Comparable data, ComparableLink next){
                this.data = data;
                this.next = next;
        }
        public ComparableLink (Comparable data){
                this(data,null);
        }
}//class ComparableLink
```

שימו לב לא להשתמש בשיטות של המחלקה Link . **לתזכורת**, נתון לכם הממשק Comparable כפי שנלמד בכיתה:

```
public interface Comparable{
        public int compareTo(Object other);
}
```

השיטה compareTo תופעל על האובייקט this ותיבחן את היחס בינו ובין other. הדרישות מהשיטה compareTo היא להחזיר מספר שלילי אם this יותר קטן, 0 במקרה של שוויון ומספר חיובי אם other יותר קטן.
נתונה לכם המחלקה SortedList הבאה:

```
public class SortedList{
        private ComparableLink sentinel;

        public SortedList(){
                sentinel = new ComparableLink("sentinel");
        }
        public boolean isEmpty(){
                return sentinel.next == null;
        }
        public boolean isSorted(){
                if (this.isEmpty())
                        return true;
                return sentinel.next.isSorted();
        }
}//class SortedList
```

בשאלה זו, תצטרכו להוסיף שיטה אחת למחלקה SortedList (סעיף א'), ושיטה אחת למחלקה ComparableLink (סעיף ב').

המשך בעמוד הבא

<u>סעיף א' (10 נקודות)</u>

כיתבו שיטה במחלקה SortedList אשר מוסיפה אובייקט מסוג Comparable לרשימה תוך שמירה על סדר האיברים ברשימה. כלומר, לאחר הוספת האיבר החדש לרשימה, הרשימה עדיין נשארת ממוינת.

```
public void add(Comparable data){
        השלימו בדף התשובות//
}
```

<u>סעיף ב' (10 נקודות)</u>

השיטה isSorted במחלקה SortedList מחזירה true אם ורק אם הרשימה ממוינת. השיטה קוראת לשיטה isSorted במחלקה ComparableLink. עליכם לממש את השיטה isSorted במחלקה ComparableLink אשר מחזירה true אם ורק אם הרשימה אשר מתחילה בעצם המפתח (this) ממוינת:

```
public boolean isSorted(){
        השלימו בדף התשובות//
}
```

שאלה 5 (20 נקודות)

בשאלה זו 5 סעיפים (א-ה), אותם יש למלא בדף התשובות. בדף התשובות תמצאו סעיפים המתיחסים לקוד הבא, הכולל את המחלקות Person ו-Student, וכן המחלקה Run הכוללת main:

```
public class Person {
	private String name;
	private int id;
	public String className;

	public Person(String name, int id) {
		this.name = name;
		this.id = id;
		className = "Person";
	}

	public String getName() {return name;}
	private int getId() {return id;}

	public void printClassName1() {System.out.println(className);}
	public void printClassName2() {System.out.println(className);}

	public void printName() {System.out.println("Name: " + getName());}
	public void printId() {System.out.println("Id: " + getId());}
} // class Person

public class Student extends Person {
	public String className;

	public Student(String name, int id) {
		super(name, id);
		className = "Student";
	}

	public String getName() {return name + " (student)"; }
	public int getId() {return 1111;}

	public void printClassName2() {System.out.println(className);}
} //class Student

public class Run {
	public static void main(String[] args) {
	Person p = new Student("Moti", 1234);
	  <a>
}
```

בהצלחה

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 $0 \leq r < 1$,r