מבוא למדעי המחשב
202-1-101-1

מבוא לתכנות למערכות מידע
202-1-104-1

סמסטר א' תשס"ה
<u>בוחן שני – 10.12.04</u>

דר' ג'יהאד אל-סנע
גב' דיקלה דותן
מר ענב וינרב
פרופ' מיכאל קודיש
דר' יצחק רוזן

משך הבוחן שעתיים וחצי (לא תינתן הארכה)
חומר עזר - אסור
אין להשתמש במחשבון.

במבחן זה 4 שאלות בניקוד המסתכם ב-100 נקודות. ענו על כל השאלות.

אנא רשמו את תשובותיכם בדף התשובות בלבד. המחברת שקיבלתם היא מחברת טיוטה והיא לא תימסר כלל לבדיקה. בסיום הבחינה נשמור אך ורק את דף התשובות. כל שאר החומר יועבר לגריסה.
הקפידו לרשום בדף התשובות גם את מספר הנבחן ומספר החדר שבו אתם נבחנים.

בשאלות התכנות, מספר השורות העומדות לרשותכם בדף התשובות רומז על אורך הקוד הנדרש. הקפידו על כתב יד ברור. תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא. אין צורך להעתיק את שורות הקוד הנתונות בשאלון לדף התשובות. מותר להסתמך על סעיפים קודמים גם אם לא עניתם עליהם.

בדף האחרון של המבחן ישנה תזכורת למספר פונקציות חשובות, שתוכלו לעשות בהן שימוש במהלך המבחן.

שימו לב: בשאלות התכנות החשיבות העליונה היא לנכונות הקוד. מאידך, יעילות וסגנון חשובים גם הם, ולכן תשובה יעילה ומסוגננת תזכה בציון גבוה יותר. בשאלות בהן לא מצוין במפורש שיש לכתוב פונקציה רקורסיבית, תוכלו לכתוב פונקציה איטרטיבית או רקורסיבית, לבחירתכם.

בהצלחה!

שאלה 1 (40 נקודות)

בשאלה זו חמישה סעיפים.
ניתן להשתמש בסעיפים קודמים גם במידה ולא פתרתם אותם.

סעיף א' (8 נקודות)

השלימו בדף התשובות את השיטה הבוליאנית isSingle(String[] array, int i), המחזירה ערך true אם ורק אם האיבר במקום ה-i במערך array, שונה בתוכנו מכל האיברים במקומות i+1 ואילך.
הניחו בשאלה זו כי array ≠ null וכי הפרמטר i הינו אינדקס חוקי במערך array,
כלומר: array.length > i ≥ 0 (אין צורך לבדוק זאת).

```
public static boolean isSingle(String[] array, int i) {
     boolean answer = true;

     for (int j=i+1; j<array.length & answer; j=j+1)
          if (array[i] == null)
               answer = array[j] != null;
          else
               answer = ! array[i].equals(array[j]);

     return answer;
}
```

לדוגמא:

String [] a = {“abc”, “ab”, “def”, “a”, “ab”};
isSingle(a,0) → true
isSingle(a,1) → false

סעיף ב' (10 נקודות)

השיטה הבוליאנית allDiff(String[] a) הנתונה, מחזירה ערך true אם ורק אם כל איברי המערך a הינם שונים (בתוכנם).
השלימו בדף התשובות את שיטת העזר **הרקורסיבית** allDiff(String[] a, int i), הבודקת שכל האיברים החל מהאיבר ה-i (כולל) הינם שונים. הניחו בשאלה זו כי a ≠ null (אין צורך לבדוק זאת).

```
public static boolean allDiff(String[] a) {
     return allDiff(a,0);
}

public static boolean allDiff(String[] a, int i) {

     if (i == a.length-1)
          return true;
     else
          return (allDiff(a,i+1) & isSingle(a,i));

}
```

<u>סעיפים ג' ו-ד' (5 נקודות כ"א)</u>

השלימו בדף התשובות את השיטה **הרקורסיבית** grayPair(String s1, String s2), המחזירה ערך true אם ורק אם המחרוזות s1 ו-s2 הינן מחרוזות בינאריות, שונות מ-null, בעלות אורך זהה, ונבדלות בביט אחד בלבד.

```
public static boolean grayPair(String s1, String s2) {
     /*
     בדיקה שהמחרוזות אינן null, שאורכן גדול מ 0
     ושהתו הראשון בכל אחת מהן מייצג ביט (כלומר הינו 0 או 1)
    */
     if ( s1 == null || s2 == null ||
          s1.length() == 0 || s2.length() == 0 ||
          (s1.charAt(0) != '0' & s1.charAt(0) != '1') ||
          (s2.charAt(0) != '0' & s2.charAt(0) != '1') )
          return false;

     if (s1.charAt(0) == s2.charAt(0)){
          return grayPair(s1.substring(1),s2.substring(1));
    }
     else {
          return s1.substring(1).equals(s2.substring(1));
     }
}
```

<u>שימו לב</u>: שאלה זו שגויה. קוד זה מחזיר true בהרצת grayPair("0abc","1abc"), אף על פי שאלו אינן מחרוזות בינאריות.

פתרון מתחכם לשאלה זו:

```
public static boolean grayPair(String s1, String s2) {
     /*
     בדיקה שהמחרוזות אינן null, שאורכן גדול מ 0
     ושהתו הראשון בכל אחת מהן מייצג ביט (כלומר הינו 0 או 1)
    */
     if ( s1 == null || s2 == null ||
          s1.length() == 0 || s2.length() == 0 ||
          (s1.charAt(0) != '0' & s1.charAt(0) != '1') ||
          (s2.charAt(0) != '0' & s2.charAt(0) != '1') )
          return false;

     if (s1.charAt(0) == s2.charAt(0)){
          return grayPair(s1.substring(1),s2.substring(1));
    }
     else {
          return (s1.substring(1).equals(s2.substring(1)) &&
            (s1.length()==1 ||
            grayPair(s1.substring(1)+"0", s2.substring(1)+"1")));
    }
}
```

סעיף ה' (12 נקודות)

קוד Gray באורך n, הינו סידור של כל 2^n המחרוזות הבינאריות באורך n, כך שבכל מעבר ממחרוזת לבאה אחריה בסידור, יש שינוי רק בביט (סיבית) אחד (כולל במעבר מהמחרוזת האחרונה לראשונה).

למשל, בטבלה הבאה נתונים 3 קודי Gray שונים, כאשר n=3 :

000	111	000
010	101	001
011	100	011
001	000	010
101	001	110
111	011	111
110	010	101
100	110	100

השלימו בדף התשובות את השיטה isGray(String[] a), המחזירה ערך true אם ורק אם המערך a מכיל קוד Gray חוקי. הניחו בשאלה זו כי המערך a ≠ null וכי אורכו גדול מ-1 (אין צורך לבדוק זאת).

```
public static boolean isGray(String[] a) {
     boolean ans;
     ans = ((a[0] != null) &&
           ((int)Math.pow(2,a[0].length()) == a.length) &
           (allDiff(a)));
     for (int i=0; i<a.length & ans; i=i+1)
            if (! grayPair(a[i],a[(i+1)%a.length])
                  ans = false;
     return ans;
}
```

שאלה 2 (16 נקודות)

בשאלה זו שלושה סעיפים. יש לסמן תשובה נכונה אחת בלבד בכל סעיף.

נתונות הפונקציות הבאות:

```
public static void rec0(String s) {
	if (s != null) {
		rec1(s,"");
	}
}

public static void rec1(String s1, String s2){
	if (s1.length()>0) {
		rec1(delete(s1,0),s2);
		rec2(delete(s1,0),s2+(s1.charAt(0)));
	}
}

public static void rec2(String s1, String s2) {
	if (s2.length()>0)
		System.out.println(s2);
	if (s1.length()>0)
		rec2(delete(s1,0),s2+(s1.charAt(0)));
}

public static String delete(String s, int i){
	return(s.substring(0,i)+s.substring(i+1,s.length()));
}
```

סעיף א' (5 נק')

בהרצת rec0("ab") יודפס למסך:

א. **b** **a** **ab**	ב. b a	ג. b a ba ab	ד. a b ab	ה. ba a b	ו. a b	ז. ab ba a b

הגדרות (לסעיפים הבאים):

בהנתן מחרוזת s -

- תת מחרוזת (לא ריקה) של s הינה קבוצה סדורה של אותיות, המופיעות בסדר זהה במחרוזת s.
  למשל, תתי המחרוזות (הלא ריקות) עבור s ="abc" הינן:
  "abc", "bc", "ac", "ab", "c", "b", "a".
- תת מחרוזת (לא ריקה) רציפה הינה קבוצה סדורה של אותיות, המופיעות ברצף ובסדר זהה במחרוזת s. למשל, תתי המחרוזות הרציפות (הלא ריקות) עבור s ="abc" הינן:
  "abc", "bc", "ab", "c", "b", "a".
- תת מחרוזת תחילית הינה תת מחרוזת לא ריקה רציפה המתחילה בתו הראשון של המחרוזת s.
  למשל, תתי המחרוזות התחיליות עבור s ="abc" הינן: "abc", "ab", "a".
- תת מחרוזת סופית הינה תת מחרוזת לא ריקה רציפה המסתיימת בתו האחרון של המחרוזת s.
  למשל, תתי המחרוזות הסופיות עבור s ="abc" הינן: "abc", "bc", "c".

סעיף ב' (5 נק')

מה מדפיסה הפונקציה `rec2` בעת קריאה ל- `rec2(s,"")` בהנחה ש `s≠null` וש `s≠""`?

א. יודפסו בסדר כלשהו כל תתי המחרוזות הלא ריקות של s.
ב. יודפסו בסדר כלשהו כל תתי המחרוזות הלא ריקות של s, כל אחת מהן מהסוף להתחלה.
ג. יודפסו בסדר כלשהו כל תתי המחרוזות הרציפות הלא ריקות של s.
ד. יודפסו בסדר כלשהו כל תתי המחרוזות התחיליות הלא ריקות של s.
ה. יודפסו בסדר כלשהו כל תתי המחרוזות הסופיות הלא ריקות של s.
ו. יודפסו בסדר כלשהו כל התוים של s.
ז. יודפסו בסדר כלשהו כל התוים של s, וכן המחרוזת הריקה.

סעיף ג' (6 נק')

מה מדפיסה הפונקציה `rec0` בעת קריאה ל- `rec0(s)` בהנחה ש `s≠null` וש `s≠""`?

א. יודפסו בסדר כלשהו כל תתי-המחרוזות הלא ריקות של s.
ב. יודפסו בסדר כלשהו כל תתי המחרוזות הלא ריקות של s, כל אחת מהן מהסוף להתחלה.
ג. יודפסו בסדר כלשהו כל תתי המחרוזות הרציפות הלא ריקות של s.
ד. יודפסו בסדר כלשהו כל תתי המחרוזות התחיליות הלא ריקות של s.
ה. יודפסו בסדר כלשהו כל תתי המחרוזות הסופיות הלא ריקות של s.
ו. יודפסו בסדר כלשהו כל התוים של s.
ז. יודפסו בסדר כלשהו כל התוים של s, וכן המחרוזת הריקה.

שאלה 3 (24 נקודות)

תשובה נכונה על כל אחד משמונת הסעיפים בשאלה זו, מזכה ב-3 נקודות.
תשובה שגויה בסעיף מורידה נקודה אחת.
אי סימון בסעיף מסוים לא יגרור הורדת נקודה.
סך הנקודות בשאלה לא ירד מ-0.
יש לסמן תשובה נכונה אחת בלבד לכל סעיף.

נתונות המחלקות הבאות:

```
class Student {
    private String name;
    private int id;

    public Student(int id, String name) {
        this.name = new String(name);
        this.id = id;
    }
    public String getName() {
       return name;
    }
    public boolean equals(Object other) {
       return ((other!=null) && (other instanceof Student) &&
              (this.id == ((Student)other).id));
    }
```

```
}
class Tomato{
    private int weight;

    public Tomato(int weight){
        this.weight = weight;
    }
    public int getWeight(){
        return this.weight;
    }
    public boolean equals(Tomato other) {
        return ((other!=null) && (this.weight == other.weight));
    }
}


class Pepper{
    public int weight;

    public Pepper(int x){
        this.weight = x;
    }
    public int getWeight(){
       return this.weight;
    }
    public boolean equals(Object other){
        return ((other!=null) && (other instanceof Pepper) &&
               (this.weight == ((Pepper) other).weight));
    }
}
```

בנוסף, נתון הקוד הבא:

```
class Question3 {
      public static void main (String [] args) {
            Student x1 = new Student(5050, “Avi");
            Object  x2 = new Student(5050, “Dan”);

            Tomato y1 = new Tomato(10);
            Object y2 = new Tomato(10);

            Pepper z1 = new Pepper(10);
            Object z2 = new Pepper(10);
        }
}
```

א) `System.out.println(x2.name);`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה

ב. שגיאה בזמן ריצה

ג. הדפסת "10" למסך

ד. הדפסת "0" למסך

ה. הדפסת פלט אחר למסך

ב) `System.out.println(((Student)y2).getName());`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה

ב. שגיאה בזמן ריצה

ג. הדפסת "10" למסך

ד. הדפסת "Dan" למסך

ה. הדפסת פלט אחר למסך

ג) `System.out.println(z2.weight==y1.weight);`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה

ב. שגיאה בזמן ריצה

ג. הדפסת "true" למסך

ד. הדפסת "false" למסך

ה. הדפסת פלט אחר למסך

ד) `System.out.println(y2.equals(y1));`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה

ב. שגיאה בזמן ריצה

ג. הדפסת "true" למסך

ד. הדפסת "false" למסך

ה. הדפסת פלט אחר למסך

ה) `System.out.println(x2.equals(x1));`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה

ב. שגיאה בזמן ריצה

ג. הדפסת "true" למסך

ד. הדפסת "false" למסך

ה. הדפסת פלט אחר למסך

ו) `System.out.println(x1.equals((Student)x2));`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה

ב. שגיאה בזמן ריצה

ג. הדפסת "true" למסך

ד. הדפסת "false" למסך

ה. הדפסת פלט אחר למסך

ז) `System.out.println(z1.equals((Tomato)y2));`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה
ב. שגיאה בזמן ריצה
ג. הדפסת "true" למסך
ד. הדפסת "false" למסך
ה. הדפסת פלט אחר למסך

ח) `System.out.println(z1.equals((Pepper)y2));`

הוספת שורה זו, תגרום ל-

א. שגיאת קומפילציה
ב. שגיאה בזמן ריצה
ג. הדפסת "true" למסך
ד. הדפסת "false" למסך
ה. הדפסת פלט אחר למסך

שאלה 4 (20 נקודות)

נתון הממשק ObjectSetInterface:

```
public interface ObjectSetInterface {
      public boolean isEmpty();  // בודק האם הקבוצה ריקה
      public boolean contains(Object t); // בודק האם האיבר בקבוצה
      public void add(Object t); // הוספת איבר לקבוצה
      public void remove(Object t); // הוצאת איבר מהקבוצה
      public Object[] toArray();
       // מחזיר מערך של כל איברי הקבוצה (לא העתקים של האיברים)
}
```

נתונה גם המחלקה class ObjectSet implements ObjectSetInterface {…}
המממשת את הממשק הנתון. פרטי המימוש אינם ידועים לכם.
עליכם להגדיר שתי פונקציות נוספות עבור המחלקה ObjectSet, תוך שימוש בשיטות הקיימות במחלקה, אך ללא התייחסות לצורת מימושן של שיטות אלו.

סעיף א' (10 נקודות)

ממשו את השיטה:

```
public void intersection (ObjectSetInterface other)
```

לאחר הקריאה לשיטה זו, יהווה עצם המפתח את חיתוך הקבוצות של קבוצת עצם המפתח והקבוצה other.
הניחו כי other≠null.
למשל, חיתוך קבוצות {a,b,c,d} ו-{a,c,g,h,f} הינו {a,c}

```
Object [] a = other.toArray();
for( int i=0; i<a.length & ! this.isEmpty(); i=i+1)
      if (! this.contains(a[i])
            this.remove(a[i]);
```

סעיף ב' (10 נקודות)

ממשו את השיטה:

```
public boolean isSubset (ObjectSetInterface other)
```

שיטה זו מחזירה true במידה והקבוצה other הינה תת קבוצה של עצם המפתח. הניחו כי other≠null.
למשל, אם A מייצגת את הקבוצה {a,b,c,d} ו-B מייצגת את הקבוצה {b,c} אזי:
A.isSubset(B) → true
B.isSubset(A) → false

```
Object [] a = other.toArray();
boolean ans = true;
for (int i=0; i<a.length & ans; i=i+1)
     if (! this.contains(a[i])
          ans = false;
return ans;
```

תזכורת:

- `char` **`charAt`**`(int index)`

  שיטה במחלקה String, המחזירה את התו במיקום index במחרוזת

- `int` **`length`**`()`

  שיטה במחלקה String המחזירה את אורך המחרוזת

- `String` **`substring`**`(int beginIndex, int endIndex)`

  שיטה במחלקה String המחזירה מחרוזת חדשה, שהיא תת-מחרוזת של עצם המפתח בין התווים beginIndex ו-endIndex, (כולל התו ה-beginIndex ולא כולל התו ה-endIndex)

- `String` **`substring`**`(int beginIndex)`

  שיטה במחלקה String המחזירה מחרוזת חדשה, שהיא תת-מחרוזת של עצם המפתח החל מהתו beginIndex (כולל) ועד לסוף המחרוזת.

- `double` **`Math.pow`**`(double a, double b)`

  פונקציה סטטית המחזירה את הערך של a בחזקת b

- `double` **`Math.sqrt`**`(double a)`

  פונקציה סטטית המחזירה את השורש הריבועי החיובי של a