תאריך הבחינה: 14.2.2013

שמות המרצים: פרופ' משה זיפר
מר אילן כדר
ד"ר פז כרמי
מר עדי סוויסה
פרופ' מייק קודיש
ד"ר צחי רוזן

שם הקורס: מבוא למדעי המחשב

מספר הקורס: 202-1-1011

שנה: 2013 סמסטר: א' מועד: א

משך הבחינה: **2.5 שעות**

חומר עזר: **אסור**

מבוא למדעי המחשב **מועד א**

אנא קיראו היטב את ההוראות שלהלן:

- במבחן זה 4 שאלות. ענו על כל השאלות. בשאלות בהן לא מצוין אחרת, ניתן לבחור בפתרון רקורסיבי או פתרון שאינו רקורסיבי, לבחירתכם.
- רשמו את תשובותיכם **בדפי 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.
- **שימו לב:** החשיבות העליונה היא **לנכונות** הקוד. מאידך, **יעילות, סגנון וכתיבה ברורה** חשובים גם הם, ולכן תשובה יעילה ומסוגננת תזכה בציון גבוה יותר.
- בשאלות התכנות, מספר השורות העומדות לרשותכם בדף התשובות רומז על אורך הקוד המקסימלי הנדרש. הקפידו על כתב יד ברור. **תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא**. כתבו פתרון לשאלה קודם במחברת הטיוטה ורק לאחר מכן העתיקו פתרון נקי לדף התשובות.
- בכל השאלות, אין להוסיף פונקציות עזר אלא אם נאמר במפורש שמותר.
- אם יש סעיף בשאלה המסתמך על סעיף אחר, מותר להשתמש בו גם אם לא פתרתם את הסעיף האחר.
- **הקפידו לרשום בכל דפי התשובות גם את מספר הנבחן ומספר החדר שבו אתם נבחנים.**
- **במידה ואינכם יודעים את התשובה לסעיף כלשהו, רשמו "לא יודע/ת" (במקום תשובה) ותזכו ב- 20% מניקוד הסעיף (מעוגל מטה). אם רשום "לא יודע/ת", ההתייחסות היא לכל הסעיף.**

<u>שאלה 1</u> (25 נקודות) בשלושת הסעיפים הבאים יש לתת פתרון **יעיל גם בזמן ריצה וגם בזכרון.** ניתן להשתמש בסעיף קודם גם אם לא פתרתם אותו. בכל הסעיפים ניתן להניח כי $n > 0$.

<u>סעיף א (6 נקודות)</u> נגדיר סדרה בשם sir:

$$sir(n) = \begin{cases} 0 & n = 1 \\ 0 & n = 2 \\ 1 & n = 3 \\ sir(n-1) + sir(n-2) + sir(n-3) & n > 3 \end{cases}$$

האיברים הראשונים בסדרה (החל מ n=1) הם: 0, 0, 1, 1, 2, 4, 7, 13, 24, 44 ...

כתבו פונקציה `int sir(int n)` אשר מחשבת את האיבר ה `n` -י בסדרה `sir`.

<u>סעיף ב (4 נקודות)</u> פז הגדיר סדרה חדשה, דומה לקודמת:

$$p3(n) = \begin{cases} 0 & n = 1 \\ 0 & n = 2 \\ 1 & n = 3 \\ p3(n-1) + p3(n-2) + p3(n-3) + \mathbf{1} & n > 3 \end{cases}$$

האיברים הראשונים בסדרה `p3` הם: 0, 0, 1, 2, 4, 8, 15, 28, ...

כתבו פונקציה `int p3(int n)` אשר מחשבת את האיבר ה `n` -י בסדרה `p3`.

סעיף ג (15 נקודות)

פז מימש את p3 באופן הבא:

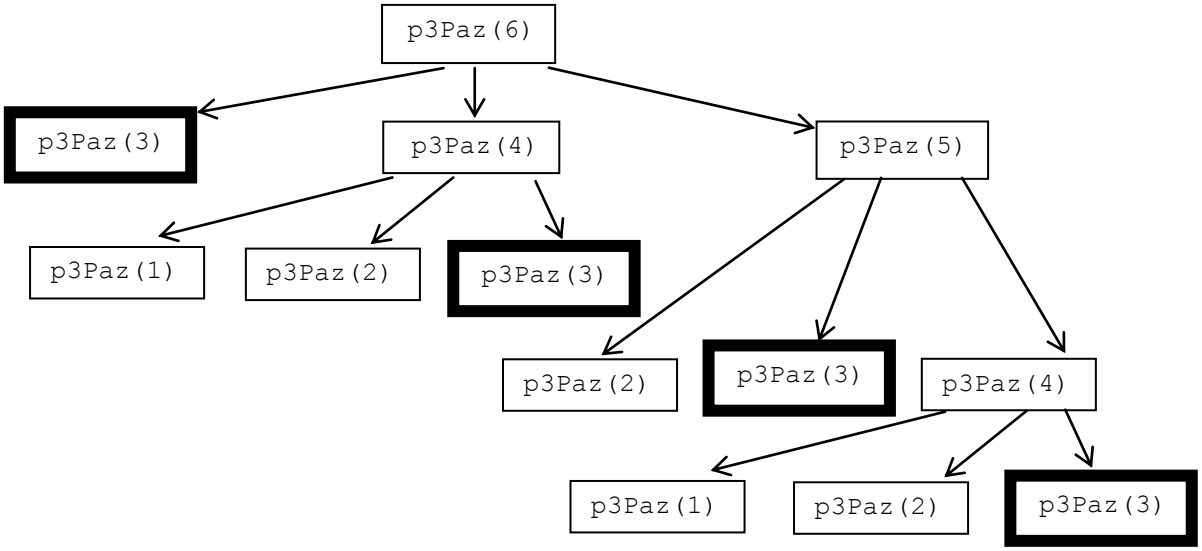
```
public static int p3Paz(int n) {
     int ans;
     if (n==1||n==2)
          ans = 0;
     else if (n==3)
          ans = 1;
     else
          ans = p3Paz(n-1) + p3Paz(n-2) + p3Paz(n-3) + 1;
     return ans;
}
```

הוא הבין שהקוד שלו לא יעיל, בעיקר אחרי שראה את הפתרון שלכם (בסעיף ב), ותהה לעצמו כמה קריאות (חוזרות) יש במהלך הריצה במימוש שלו. לצורך כך הוא שאל את עצמו כאשר הוא מריץ את הפונקציה p3Paz(n) (שימו לב: p3Paz, לא p3 ולא sir) כמה פעמים נקראת הפונקציה p3Paz(m) כאשר $2 < m < n$.

עזרו לפז במציאת הפתרון ע"י כתיבת הפונקציה int pazCount(int n, int m) אשר עבור n, m $(2 < m < n)$ מחזירה את מספר הקריאות לפונקציה p3Paz(m) בחישוב של p3Paz(n).

למשל אם $n = 6$ אזי יש קריאה אחת ל- p3Paz(5) וארבע קריאות ל- p3Paz(3). כלומר, הקריאה ל- pazCount(6,5) תחזיר 1 והקריאה ל- pazCount(6,3) תחזיר 4 כמודגם בציור בתחתית העמוד.

יש להניח שהקלט תקין. כלומר, $2 < m < n$.



<u>שאלה 2 (25 נקודות)</u>

בשאלה זו נשתמש בשתי מחלקות: Pixel ו-Painter על מנת לצייר רחוב עם בניינים.

נתונה המחלקה **Pixel** המתארת פיקסל במישור התמונה באמצעות שני ערכים שלמים x ו- y (המחלקה מצורפת בסוף השאלה).

המחלקה **Painter** מכילה את הפונקציה הבאה המציירת קו ישר בין שני הפיקסלים p1=(x1,y1) ו-p2=(x2,y2).

```
public static void drawLine(Pixel p1, Pixel p2)
```

בשאלה זו **מותר וצריך** להשתמש בפונקציה העזר drawLine, אך לא תצטרכו פרטים נוספים על המחלקה Painter לצורך מתן תשובה.

בשאלה זו אתם נדרשים להשלים את הפונקציה **הרקורסיבית** שמציירת רחוב בעומק depth בין הפיקסלים p1 ו-p2.

```
public static void draw(Pixel p1, Pixel p2,int depth){

    //השלמו בדף התשובות

}
```

כאשר:

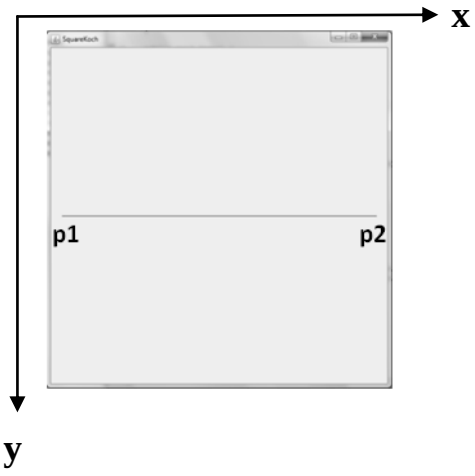
p1 הינו פיקסל המגדיר את תחילת הרחוב– הנקודה השמאלית ביותר של הרחוב.

p2 הינו פיקסל המגדיר את סוף הרחוב– הנקודה הימנית ביותר של הרחוב.

depth– הינו עומק הרחוב – תפקידו יובהר בדוגמאות הבאות.

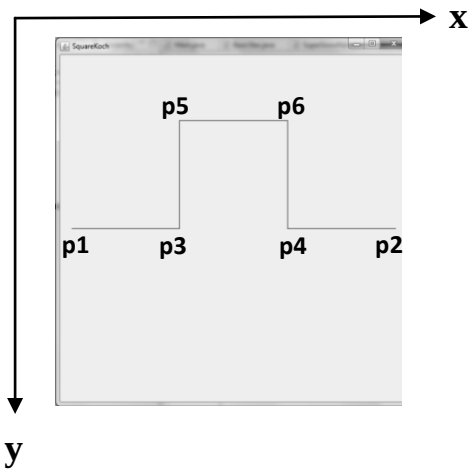
<u>רחוב בעומק 0 (depth=0)</u>

רחוב זה מכיל קו רציף בין הפיקסלים p1 ל-p2 ואינו מכיל בניינים (ראו תמונה).



רחוב בעומק 1 (depth=1)

על הרחוב להכיל בניין יחיד הממוקם במרכז הרחוב. רוחבו וגובהו של הבניין שווים לשליש מאורך הרחוב (ראו תמונה).

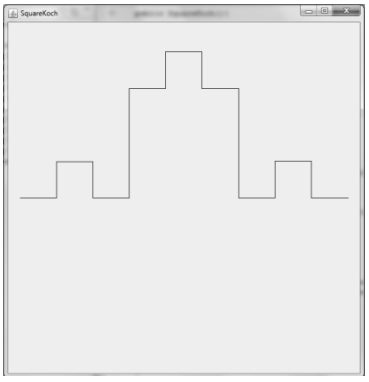


שימו לב שכל המקטעים בתמונה שווים באורכם, ושאורכו של כל מקטע שווה לשליש מאורך הרחוב. כלומר

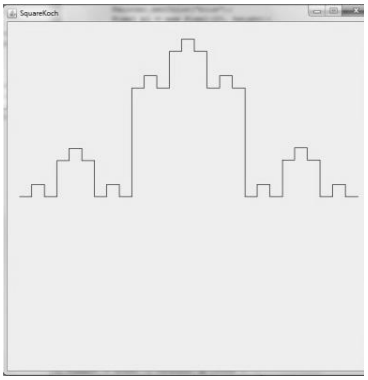
$$(p3.x-p1.x) = (p3.y-p5.y) = (p6.x-p5.x) = (p4.y-p6.y) = (p2.x-p4.x) = ((p2.x-p1.x)/3)$$

רחוב בעומק 2 (depth=2)

ברחוב זה על הבניין האמצעי נוסף מגדל. המגדל הוא ריבוע, שצלעותיו באורך של שליש מרוחב הבניין האמצעי ומיקומו במרכז הגג של הבניין האמצעי.
ברחוב זה נוספו גם שני בניינים קטנים מימין ומשמאל לבניין האמצעי. בניינים אלו הינם ריבועים וגודלם (רוחב וגובה) כגודלו של המגדל ומיקומם במרכז המקטעים הימני והשמאלי בהתאמה.



עתה, אם נמשיך לבנות באותו אופן, עבור depth=3 נקבל את הרחוב הבא:



<u>שימו לב:</u> יש להניח עבור משתני הקלט:

- p1 ו-p2 הינם ערכים תקינים, כלומר אינם null ואינם חורגים מגבולות החלון הגרפי.
- p1.getX() < p2.getX()
- p1.getY() == p2.getY()
- $depth \geq 0$

```
public class Pixel{

	private int x;
	private int y;

	public Pixel(){
		x=0;
		y=0;
	}
	public Pixel(int x, int y){
		this.x=x;
		this.y=y;
	}
	public int getX(){
		return x;
	}

	public int getY(){
		return y;
	}

}
```

שאלה 3 (25 נקודות)

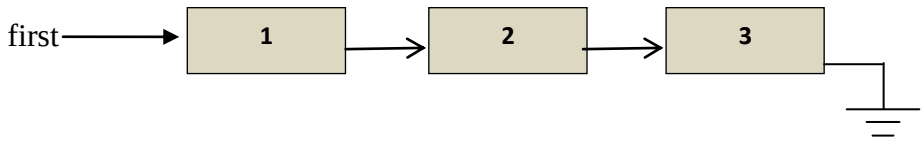
בשאלה זו נתייחס לרשימות מקושרות שבחוליות שלהן יש ערכים שלמים.
נתונה לכם המחלקה IntLink הבא המתארת חוליה ברשימה המקושרת:

```
public class IntLink {

public int data;
      public IntLink next;

      public IntLink(int data, IntLink next) {
            this.data = data;
            this.next = next;
      }
      public IntLink(int data) {
            this(data, null);
      }
}
```

שימו לב: בשאלה זו נגדיר רשימה מקושרת ע"י מצביע מטיפוס IntLink לחוליה הראשונה ברשימה (המצביע first בדוגמא הבאה).

לדוגמא, הקוד הבא יוצר את הרשימה המקושרת שבתמונה למטה:

```
IntLink third = new IntLink(3);
IntLink second = new IntLink(2,third);
IntLink first = new IntLink(1,second);
```



<u>סעיף א' (12 נק')</u>

נתון הממשק הבא:

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

מחלקה המממשת את הממשק Filter מאפשרת להבחין בין עצמים לפי קריטריון שבחרנו מראש.
לדוגמא: המחלקה IntLinkFilter המממשת את הממשק Filter מאפשרת להבחין בין עצמים מטיפוס IntLink לבין כל שאר העצמים.

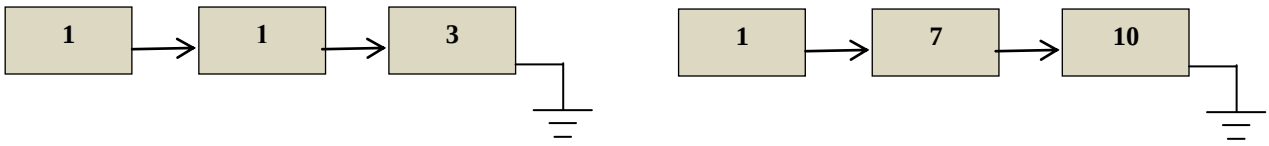
```
public class IntLinkFilter implements Filter {
     public boolean accept(Object obj) {
          return (obj instanceof IntLink);
     }
}
```

עליכם להשלים בדף התשובות את המחלקה SortedIntLinkFilter המממשת את הממשק Filter ומאפשרת להבחין בין רשימות מקושרות מטיפוס IntLink **הממוינות בסדר עולה** לבין כל עצם אחר. "סדר עולה" משמעו כי כל איבר ברשימה גדול או שווה לאיבר הקודם לו ברשימה.

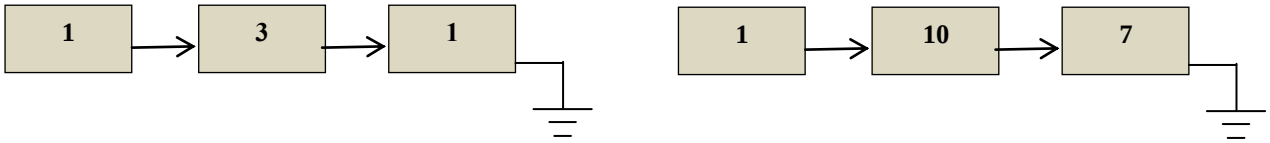
הנחה: רשימה באורך 1 מקיימת את הקריטריון של הפילטר.
ניתן להניח שהקלט המתקבל הינו אובייקט מטיפוס IntLink המצביע לחוליה הראשונה ברשימה (כמו המשתנה first מהדוגמא בעמוד הקודם).

```
public class SortedIntLinkFilter implements Filter {
     public boolean accept(Object obj){
          //השלמו בדף התשובות
     }
}
```

שתי דוגמאות לרשימות שעומדות בקריטריון:



שתי דוגמאות לרשימות **שלא** עומדות בקריטריון:



<u>סעיף ב' (13 נק')</u>

נתונים הממשקים Iterator ו-Array אותם ראינו בכיתה:

```
public interface Iterator{
  // returns true if the iteration has more elements
  public boolean hasNext();

  // returns the next element in the iteration
  //throws NoSuchElementException if has no more elements
  public Object next();
}

public interface Array{
     public Object get(int i);
     public void set(int i, Object data);
     public int size();
}
```

עליכם להשלים בדף התשובות את המחלקה ArrayOfIntLinkIterator. מחלקה זו מממשת איטרטור עבור מערך של רשימות מקושרות ומאפשרת מעבר איטרטיבי אך ורק על הרשימות במערך **המקיימות** קריטריון שהוגדר באובייקט מטיפוס Filter. עליכם להשלים את בנאי המחלקה המקבל מערך של רשימות מקושרות, את מספר הרשימות המקושרות במערך וכן אובייקט מטיפוס Filter. כמו כן עליכם להשלים את השיטה hasNext כך שתחזיר true במידה וקיימת רשימה מקושרת נוספת במערך המקיימת את הקריטריון שהוגדר ב-Filter. כמו כן יש להשלים את השיטה next כך שתחזיר (במידה וקיימת) את הרשימה הבאה במערך המקיימת את הקריטריון שהוגדר ב-Filter או תזרוק שגיאת NoSuchElementException במידה ואין.

```
public class ArrayofIntLinkIterator implements Iterator {
     private Array arr;
     private int nextIndex;
     private int numOfElements;
     private Filter filter;

     public ArrayofIntLinkIterator(Array arr,
               int numOfElements, Filter filter){
          //השלמו בדף התשובות
     }

     public boolean hasNext(){
          //השלמו בדף התשובות
     }

     public Object next() {
          //השלמו בדף התשובות
     }

}
```

שאלה 4 (25 נקודות)

נתון הממשק הבא MyComparable להשוואה בין שני אובייקטים ברי השוואה:

```
public interface MyComparable{
  //return true if this is less than or equal to other
  boolean lessThanEqual(MyComparable other);

  //return true if this is less than other
  boolean lessThan(MyComparable other);

  //return true if this is greater than or equal to other
  boolean greaterThanEqual(MyComparable other);

  //return true if this is greater than other
  boolean greaterThan(MyComparable other);

  //return true if this is equal to other
  boolean equal(MyComparable other);

  //return true if this is not equal to other
  boolean notEqual(MyComparable other);
}
```

ברצוננו לקודד מחלקה המתארת נקודה במישור $p=(x,y)$ ומאפשרת להשוות בין שני עצמים מהמחלקה. על ההשוואה לתמוך ב-6 היחסים המוגדרים בשיטות של הממשק MyComparable.

שתי תלמידות הציעו דרכים שונות להשוואה בין שתי נקודות במישור:

גל הציעה שהשוואה בין שתי נקודות במישור $p_1=(x_1,y_1)$ ו-$p_2=(x_2,y_2)$ תהיה מוגדרת בצורה לקסיקוגרפית, כלומר השוואה לפי שיעור ה-x ואם ערכי ה-x שווים ההשוואה מוגדרת ע"פ שיעור ה-y. לדוגמא:

- הנקודה $p_1=(1,6)$ קטנה מהנקודה $p_2=(2,3)$ מכיוון ש- $2>1$ (היא גם קטנה שווה מכיוון ש $2\geq1$).
- הנקודה $p_1=(1,6)$ קטנה (וגם קטנה שווה) מהנקודה $p_2=(1,7)$ מכיוון שערכי ה-x של הנקודות שווים ועבור ערכי ה-y של הנקודות מתקיים ו- $6<7$.
- הנקודה $p_1=(1,6)$ שווה (וגם קטנה שווה וגם גדולה שווה) לנקודה $p_2=(1,6)$ מכיוון שגם ערכי ה-x וגם ערכי ה-y של הנקודות שווים.

ליאור הציעה שהשוואה בין שתי נקודות במישור תהיה מוגדרת ע"פ מרחק הנקודות מראשית הצירים. המרחק של של הנקודה p_1 מראשית הצירים שווה ל $\sqrt{{x_1}^2+{y_1}^2}$. לדוגמא:

- הנקודה $p_1=(1,6)$ גדולה (וגם גדולה שווה) מהנקודה $p_2=(2,3)$ מכיוון ש- $\sqrt{2^2+3^2}<\sqrt{1^2+6^2}$
- הנקודה $p_1=(1,6)$ קטנה (וגם קטנה שווה) מהנקודה $p_2=(1,7)$ מכיוון ש- $\sqrt{1^2+7^2}>\sqrt{1^2+6^2}$
- הנקודה $p_1=(2,3)$ שווה (וגם קטנה שווה וגם גדולה שווה) לנקודה $p_2=(3,2)$ מכיוון ש- $\sqrt{3^2+2^2}=\sqrt{2^2+3^2}$

<u>שימו לב:</u> על מנת לממש את ההשוואה שליאור הציעה מותר להשתמש בפונקציית העזר `public static double sqrt(double a)` של המחלקה Math.

גל וליאור רצו לממש את המחלקות GalComparablePoint ו-LiorComparablePoint המממשות את הממשק MyComparable ע"פ הגדרות ההשוואה שהציעו בהתאמה. במימוש המחלקות גל וליאור שמו לב שיש מידה רבה של קוד משותף, ועל כן החליטו להגדיר מחלקה אבסטרקטית AbsComparablePoint שתכיל את הקוד המשותף ותמנע שכפול קוד מיותר.

סעיף א' (15 נק')

השלימו בדף התשובות את המחלקה האבסטרקטית AbsComparablePoint המממשת את הממשק MyComparable.

עבור כל שיטה מהממשק MyComparable, עליכם להחליט אם לממשה או להגדירה כשיטה אבסטרקטית. על מנת להימנע משכפול קוד עליכם להגדיר **מינימום שיטות אבסטרקטיות** ולממש **מקסימום שיטות מהממשק** MyComparable במחלקה AbsComparablePoint.

```
public abstract class AbsComparablePoint implements
MyComparable {

	protected int x;
	protected int y;

	public AbsComparablePoint(int x,int y){
		this.x=x;
		this.y=y;
	}

	public int getX(){
		return this.x;
	}

	public int getY(){
		return this.y;
	}

	//השלמו בדף התשובות
}
```

<u>סעיף ב' (10 נק')</u>

השלימו בדף התשובות את המחלקות GalComparablePoint ו- LiorComparablePoint המרחיבות את המחלקה האבסטרקטית AbsComparablePoint בהתאם להגדרות ההשוואה בין שתי נקודות שהציעו גל וליאור בהתאמה.