מבוא לחישוב 6, 2-7015710-3 סמסטר ב'

מבחן מועד א' - 29.6.20

גב' אליזבת איצקוביץ, גב' רויטל מרבל

- משך המבחן: 3 שעות.
- אין שימוש בחומר עזר.
- יש להחזיר את דף המבחן בסוף המבחן.
- במבחן חמש שאלות, כולם חובה.

שאלה 1 (20 נקודות)

בשאלה זו יש לכתוב פונקציה סטטית שמקבלת מספר שלם אי-שלילי $a \geq 0$ ומחזירה מחרוזת המייצגת את התצוגה הבינארית של a.

```
public static String decimal2Binary(int a){...}
```

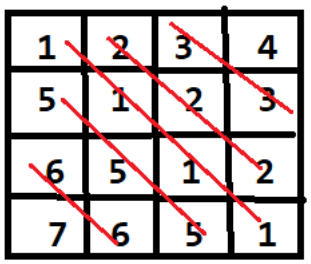
דוגמה:
קלט: $a = 6$, **פלט**: "110".

שאלה 2 (20 נקודות)

יש לכתוב פונקציה המקבלת מטריצה ריבועית של מספרים שלמים ומחזירה true במידה וכל האלכסון שהוא מקביל לאלכסון הראשי (כולל אלכסון הראשי עצמו) מורכב מאותם מספרים.

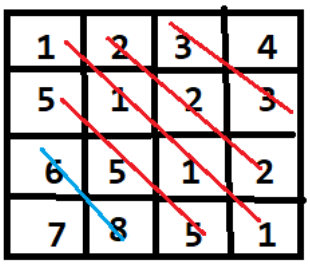
```
public static boolean sameNumbers(int[][]mat){…}
```

דוגמה 1: **פלט: true**



:

דוגמה 2 **פלט false**



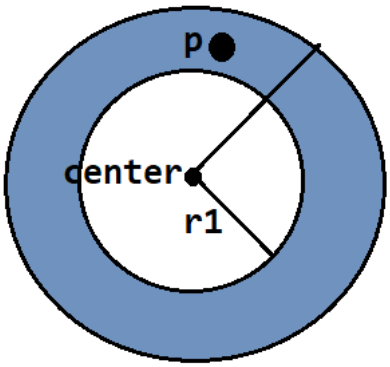
שאלה 3 (20 נקודות)

כתבו פונקציה סטטית שמקבלת שני מערכים של מספרים שלמים, ומחזירה כמה פעמים המערך **a** נמצא **ברצף** בתוך מערך **b**. ניתן להניח כי הקלט הוא תקין ו- $a.length \leq b.length$

```
public static int isIn(int[]a, int b[])
```

לדוגמה, אם הקלט הוא a[]={2,1,2}, b[]={5,1,2,3,2,1,2,1,2} הפונקציה תחזיר 2, כי מערך a נמצא פעמים במערך b, במקום 4 ו-6.

שאלה 4 (20 נקודות)



בשאלה זו יש לכתוב מחלקה: ***Ring*** המייצגת טבעת. הטבעת מוגדרת ע"י נקודת מרכז ושני רדיוסים.

יש להוסיף למחלקה שני בנאים:

- בנאי שמקבל מרכז – משתנה מטיפוס Point, ושני רדיוסים – מספרים ממשיים.
- בנאי מעתיק.

יש לכתוב פונקציה: ***int inArea(Point p)*** שמקבלת נקודה כלשהי ***p*** ומחזירה :

1 – אם הנקודה נמצאת בתוך המעגל הפנימי שרדיוסו הוא **r1,**

2 – אם הנקודה נמצאת באזור הכחול בין המעגלים כולל השפה (האזור השחור),

3 – אם הנקודה נמצאת מחוץ לטבעת.

*המחלקה Point מצורפת כנספח למבחן.

שאלה 5 (20 נקודות)

מצורפת למבחן מחלקת **LinkedListDouble** המייצגת רשימה מקושרת דו-כיוונית.
יש לכתוב פונקציה סטטית

```
public static void reverese(LinkedListDouble list)
```

הפונקציה מקבלת רשימה מקושרת דו-כיוונית והופכת אותה מהסוף להתחלה.

<u>דוגמה</u>:

קלט: [a,b,c,d,e,f], **פלט**: [f,e,d,c,b,a]

נספח: רשימה מקושרת דו-כיוונית:

```
class NodeD {
	private Object data;
	private NodeD next, prev;
	public NodeD(NodeD prev, NodeD next, Object data){
		this.data = data;
		this.prev = prev;
		this.next = next;
	}
	public String toString(){return data.toString();}
	public NodeD getNext() {return next;}
	public NodeD getPrev() {return prev;}
	public Object getData() {return data;}
	public void setNext(NodeD n) {next = n;}
	public void setPrev(NodeD p) {prev = p;}
}
public class LinkedListDouble {
	private NodeD head, tail;
	int size;
	public LinkedListDouble(){
		head = tail = null;
		size = 0;
	}
	public NodeD getHead() {return head;}
	public NodeD getTail() {return tail;}
	public void setHead(NodeD h) {head = h;}
	public void setTail(NodeD t) {tail = t;}
	public void add(Object item){
		if (head == null)
			head = tail = new NodeD(null, null, item);
		else{
			NodeD n = new NodeD(tail, null, item);
			tail.setNext(n);
			tail = n;
		}
		size++;
	}
	public String toString(){
		String s = "[";
		if (head != null){
			s = s + head.toString() + ", ";
			for (NodeD n=head.getNext();n!=null; n=n.getNext()){
				s = s + n.toString() + ", ";
			}
			s = s.substring(0, s.length()-2);
		}
		return s+"]";
	}
	public boolean contains(Object item){
		boolean ans = false;
		NodeD n = head;
		while (n != null && !n.getData().equals(item)){
			n = n.getNext();
		}
		if (n != null) ans = true;
		return ans;
	}
	public int size() {return size;}}
```

```java
/** this class represent a 2d point in the plane. <br>*/
public final class Point {
// ******** private data members ********
	private double _x, _y;  // we "mark" data members of the classusing.

// ********* constructors ********
	// initilizes the point according to its coordinates: (x1,y1)
	public Point (double x1, double y1){
			_x = x1;
			_y = y1;
	}
// initializes the point - origin (0,0)
	public Point (){
			_x = 0;
			_y = 0;
	}
//copy constructor
	public Point (Point p){
			_x = p._x;
			_y = p._y;
	}

   // ********** public methodes *********
   // returns the first coordinate of the point
   public double x() {return _x;}
   // returns the second coordinate of the point
   public double y() {return _y;}

   /** @return the distance between two points
       @parameter  p - other Point.*/
   public double distance (Point p)  {
            double temp = (p.x()-_x)*(p.x()-_x)+(p.y()-_y)*(p.y()-_y);
            return Math.sqrt (temp);
   }

   /** @return a String contains the Point data*/
   public String toString()  {
         String s = " (" + _x + "," + _y+")";
      return s;
   }

  /** logical equals:
   @param p other Point.
   @return true if p logicly the same) */
   public boolean equals (Point p) {
         boolean ans = false;
         if ((p!=null) && (p.x()==_x) && (p.y()==_y)){
               ans = true;
         }
         return ans;
   }
}// class Point
```