המרכז האוניברסיטאי אריאל

מבחן במבוא לחישוב

סמסטר א' מועד א' תשע"ג תאריך: 10.02.13

מס' קורס 7015710-2

מרצים: ד"ר ליעד גוטליב, גב' אליזבט איצקוביץ

אין להוציא את השאלון בסוף המבחן - יש להשאירו במחברת הבחינה

משך המבחן : שעתיים וחצי

ללא חומר עזר!

הוראות כלליות :

1. תשובות יש לכתוב במחברת המצורפת בלבד.
2. נא לכתוב בכתב ברור ומסודר.
3. יש לענות על בדיוק 5 שאלות מתוך 6 שאלות
   (בכל מקרה רק 5 השאלות הראשונות תיבדקנה)
   משקל כל שאלה 20 נקודות.
4. תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא.
5. אם לא נאמר אחרת ניתן להשתמש בחומר המצורף לבחינה (המחלקות נקודה, אוסף נקודות, פונקציית המיון וכו') בפתרון השאלות, מעבר לכך בהחלט ניתן לפתור שאלה בעזרת שאלה אחרת.

הקפידו על טהור הבחינה!

בהצלחה!

שאלה 1(20 נקודות)

המחלקה PointContainer שומרת אוסף של נקודות בתוך מערך פנימי. הוסיפו למחלקה שיטה שמוחקת מהאוסף את כל הנקודות הנמצאות מחוץ למעגל היחידה, כלומר מוחקת מהאוסף כל הנקודות שמרחקן עד ראשית הצירים גדול מ-1.

אחרי הקריאה לפונקציה, המשתנה sp_ של PointContainer ישמור את הגודל של האוסף החדש, ובמערך הפנימי יופיעו הנקודות ברצף ממקום 0 עד sp-1_

public void disk1(){. . . }

שאלה 2(20 נקודות)

שתי נקודות $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ נקראות נגדיות, אם $x_2 = -x_1,\ \ y_2 = -y_1$. לדוגמה, נקודות $P_1(3,-4)$, $P_2(-3,4)$ נגדיות, אבל נקודות $P_1(3,-4)$, $P_2(3,4)$ אינן נגדיות.

כתוב פונקציה סטטית שמקבלת מערך של נקודות שונות ומחזירה את מספר הזוגות של נקודות נגדיות הנמצאות במערך.

public static int opposite(Point[] pArr){. . .}

שאלה 3(20 נקודות)

נזכור מחלקת String שמייצגת מחרוזות, למחלקה יש מספר שיטות לרבות:

- int length() //מחזירה את אורך המחרוזת (מספר תווים)
- String substring(int start, int end) //מחזירה תת-מחרוזת [start,end)
- char charAt(int i) // מחזירה תו הנמצא במקום i של המחרוזת
- intcompareTo(String str) //

  מחזירה 1+ אם s>str

  מחזירה 1- אם s<str

  אחרת מחזירה אפס (שוויון מחרוזות).

כן פעולת ה + מחברת בין שתי מחרוזות.

כתוב פונקציה שמקבלת מחרוזת ובודקת האם המחרוזת מהווה מספר.

הפונקציה מחזירה "int" אם המחרוזת מהווה מספר שלם, "double" אם המספר הוא מספר ממשי, כלומר מספר שמכיל נקודה עשרונית, ומחזירה "not a number" אם המחרוזת מכילה תווים שהם לא סימן "+" או "-", לא סיפרה ולא נקודה עשרונית.

(המחרוזת יכולה להכיל סימן "+" או "-" כתו ראשון בלבד).

public static String isANumber(String s){. . . }

דוגמה קלט: s="123", פלט: "int",

קלט: s="-.123", פלט: "double"

קלט: s="12.3", פלט: "double"

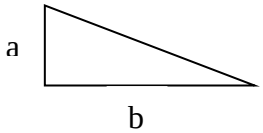
קלט: s="+123.", פלט: "double"

קלט: s="1.23.", פלט: "not a number"

קלט: s="1a23.", פלט: "not a number"

<u>שאלה 4</u>(20 נקודות)

בשאלה זו נבנה מחלקה בשם Triangle שמייצגת משולש ישר זווית. המשולש נתון על ידי אורכי הניצבים שלו: a, b (מספרים ממשיים המייצגים את אורך הצלעות).



1. כתבי את כותרת המחלקה, את השדות (המשתנים) שלה, וכן את הבנאי שמקבל את אורכי הניצבים.
2. כתבי שיטה שמחשבת ומחזירה את שטח המשולש.
3. כתבי שיטה להשוואת שני משולשים. (שני משולשים ישר זווית שווים אם הניצבים שלהם שווים). השיטה מחזירה true אם המשולשים שווים, אחרת היא מחזירה false.

<u>שאלה 5</u>(20 נקודות)

כתוב פונקציה שמקבלת מחרוזת של תווים שונים ומספר שלם n. הפונקציה מחזירה תו שמופיע במחרוזת בדיוק n פעמים. אם אין תו כזה הפונקציה תחזיר תו סימן שאלה. (אם יש כמה תווים שמופיעים n פעמים, הפונקציה תחזיר אחד מהם.)

public static char appearance(String s, int n){. . . }

<u>שאלה 6</u>(20 נקודות)

כתוב פונקציה שמדפיסה את כל המספרים הדו-ספרתיים a, ששווים לסכום שפרות של מספר a^3 .

Public static chareqA3() {…}

<u>דוגמה</u>: a=26, a^3=17576, 26= 1+7+5+7+6

<u>ספח קוד, מצורף מראש:</u>

אתם יכולים להשתמש בכל מחלקה שמוגדרת ב java.util, java,lang בפרט במחלקה Vector, כמו כן נתונה לכם הפונקציה Math.random() אשר מחזירה ערך ממשי בתחום [0,1).

חומר עזר: כולל שלוש מחלקות: נקודה, אוסף נקודות, קובץ ראשי (שמשתמש במחלקות ומדגים מיון בחירה פשוט):Point, PointContainer, Main ,

```
/** this class represents a 2d point in the plane. */
/** המחלקה מייצגת נקודה במישור. */
public class Point {
// ******** private data members תכונות פרטיות ********
        private double _x;  // we "mark" data members using _
        private double _y; // מסמנים תכונות של המחלקה ע"י מקף תחתון
```

```
// ********* constructors בנאים********
   public Point (double x1, double y1) {_x = x1; _y = y1; }
   public Point (Point p) {_x = p.x(); _y = p.y(); }

// ********** public methods שיטות פרטיות *********
   public double x() {return _x;}
   public double y() {return _y;}

   /** @return the L2 distance */
   /**הפונקציה מחזירה מרחק בין שתי נקודות*/
   public double distance (Point p)  {
      double temp = Math.pow (p.x() - _x, 2) + Math.pow (p.y() - _y, 2);
      returnMath.sqrt (temp);
   }

   /** @return a String contains the Point data*/
   /**  הפונקציה מחזירה שעורי הנקודה כמחרוזת*/
   public String toString()  {return "[" + _x + "," + _y+"]";}

   /** logical equals: return true iff p && logically the same) */
   /**  הפונקציה מחזירה "אמת" אם הנקודות שוות*/
   public boolean equals (Point p) {
      return p!=null &&p. x() == _x && p. y()==_y;
   }
}// end class Point
```

```
//////////////////////////////////// PointContainer ////////////////////////////
/** this class represent a collection of Points. */
/**המחלקה מייצגת אוסף של נקודות*/
public class PointContainer {
        // *** data members ***
        // *** תכונות המחלקה ***
        publicstatic final int INIT_SIZE=10;  // the first (init) size of the set.
                                                // ערך התחלתי של מערך נקודות
        public static final int RESCALE=10;   // the re-scale factor of this set.
                                        //  מערך נקודותערך שמשתמשים בו לשינוי גודלו של
        private int _sp=0;
        private Point[] _points;
        /*** Constructor: creates a empty set ***/
        /***בנאי ליצירת אוסף ריק***/
        public PointContainer(){
                _sp=0;
                _points = new Point[INIT_SIZE];
        }
```

```
/** returns the actual amount of Point contains in this set */
 /**  הפונקציה מחזירה מספר הנקודות הנמצאות באוסף*/
public int size() {return _sp;}

/** add a Point to this collection */
/** הוספת נקודה לאוסף*/
public void add (Point p){
        if (p != null){
                if(_sp==_points.length) rescale(RESCALE);
                _points[_sp] = new Point(p);  // deep copy semantic.
                                                // העתקה עמוקה
                //_points[_sp] = p;          // (העתקה לא עמוקה) העתקה של מצביע
                _sp++;
        }
}

/** returns a reference to the Point at the index, (not a copy) */
/** (לא עותק של נקודה)index  הפונקציה מחזירה מצביע לנקודה הנמצאת במיקום*/
public Point at(int p){
        if (p>=0 && p<size()) return _points[p];
        return null;
}

/**  check if the Container contains the point p
 * returns true if the Container contains the point potherwise returns false */
/**false הפונקציה מחזירה  trueאם נקודת pנמצאת בתוך האוסף, אחרת היא מחזירה
 */
public boolean contains(Point p){
        booleanans = false;
        for (inti=0; !ans&&i<_sp; i++){
                if(p.equals(arrPoints[i]) == true){
                        ans = true;
                }
        }
        returnans;
}
/************** private methods **************/
/************** שיטות פרטיות**************/
private void rescale(int t) {
        Point[] tmp = new Point[_sp+t];
        for(inti=0;i<_sp;i++) tmp[i] = _points[i];
        _points = tmp;
}
}//end PointContainer
```

```
//////////////////////////// main ////////////////////////////
public class Main {
	public static void main(String[] a) {
		// ****** Using Point class ********
		// ****** שימוש במחלקת נקודה********
		Point p1 = new Point(5,6),  p11=p1;
		Point p2 = new Point(1,3),  p22= new Point(p2);
		System.out.println("dist: "+p1+"  to "+p2+" is:"+p1.distance(p2));


	// ******* Using PointContainer *******
		// ******* שימוש במחלקת אוסף של נקודה *******
		PointContainer pc = new PointContainer();
		pc.add(p1);
		pc.add(p2);
		System.out.println("PointContainer: pc has "+pc.size()+" points");

		// ******* using the sort (static) function ******
		// *******מיון (פונקציה סטטית)שימוש בפונקצית******
		int[] arr = {1,4,2,6,3,9,0};
		sort(arr);
		printArray(arr);
	}
	/** selection sort: find the smallest entry, put aside ... */
	/** selection sort: מציאת המיקום של האיבר הקטן ביותר... */
	static void sort (int[] arr) {
		int to = arr.length;
		for(int from=0;from<to-1;from=from+1) {
			intminInd = minIndex(arr,from,to);
			swap(arr, from, minInd);
		}
	}
	/** @param: arr : array of integers. returns the minimum index in [from,to). */
	/** [from,to)   מחזירה מיקום של איבר מינימאלי בקטעהפונקציהמערך של שלמים,arr */
	static int minIndex(int[] arr, int from, int to) {
		intans = from;
		for(inti =from+1; i< to ; i=i+1) {
			if (arr[ans] >arr[i]) ans = i;
		}
		returnans;
	}

	/** Note: changes the array - swap i,j values. */
	/** Note:  i, j    החלפת איברים הנמצאים במקומות. */
	static void swap(int[] arr, int i, int j) {
		int tmp = arr[i];
		arr[i] = arr[j];
		arr[j] = tmp;
	}
```

```
        /** this function simply prints an array (return void). */
        /** הדפסת מערך */
        static void printArray(int[] arr) {
                System.out.println();  // new line
                for(inti =0; i<arr.length ; i=i+1) {
                        System.out.print("["+i+"]="+arr[i]+", ");  }
                System.out.println();  // new line
        }
}
```