מבוא למדעי המחשב
202-1-1011

מבוא לתכנות למערכות מידע
202-1-1041

סמסטר א' תשס"ח
מבחן מועד ב'

פרופ' מיכאל קודיש
ד"ר מיכל זיו יוקלסון
גב' דיקלה דותן
מר שחר גולן
מר טל גרינשפון

משך המבחן: שלוש וחצי שעות.
חומר עזר אסור.

במבחן זה 5 שאלות. יש לענות על כולן. הניקוד מסתכם ב 100 נקודות.

אנא רשמו את תשובותיכם בטופס הבחינה 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.

<u>מותר להסתמך על סעיפים קודמים גם אם לא עניתם עליהם.</u>

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 תשובה יעילה ומסוגננת תזכה בציון גבוה יותר.

בדף האחרון של המבחן ישנה תזכורת למספר פונקציות חשובות, בהן תוכלו להשתמש במהלך המבחן.

בהצלחה!

<u>שאלה 1 (25 נק')</u>

הממשק Filter מאפשר לנו לסנן איברים.

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

קריאה לשיטה accept(obj) תחזיר ערך בוליאני המסמן האם "לקבל" את העצם obj (במידה והערך המוחזר הוא true) או לא.

<u>סעיף א' (6 נק'):</u>

כתבו את כל המחלקה DivisibilityFilter המממשת את הממשק Filter. על הבונה במחלקה לקבל ערך מסוג int. על השיטה accept במחלקה לקבל רק עצמים מסוג Integer שערכם מתחלק בערך שקיבל הבונה. הניחו שהקריאה ל-accept היא תמיד עם עצם מסוג Integer.

```
public class DivisibilityFilter implements Filter {
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
}
```

<u>סעיף ב' (6 נק'):</u>

כתבו את כל המחלקה MajorityFilter המממשת את הממשק Filter. הבונה במחלקה מקבל שלושה עצמים מסוג Filter. השיטה accept של המחלקה תקבל אך ורק אובייקטים אותם מקבלים לפחות שניים מהפילטרים שהתקבלו בבונה.

```
public class MajorityFilter implements Filter {
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
}
```

עבור הסעיפים ג-ד נתונה המחלקה List המכילה שיטה

```
public void addLast(Object data)
```

המוסיפה את האובייקט data לחוליה חדשה בסוף הרשימה (אין צורך לממשה).

סעיף ג' (7 נק'):

בנספח המצורף בסוף המבחן ניתן למצא את הקוד של הממשקים Iterable ו- Iterator .

השלימו את הפונקציה הבאה המקבלת Filter ו- Iterable ומחזירה רשימה **חדשה** המכילה אך ורק את האיברים מתוך ה Iterable שהפילטר מקבל. שימו לב: יש לשמור על סדר האיברים המקורי.

```
public List filter(Iterable ds, Filter f){

      List ans=new List();
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
    return ans;
}
```

סעיף ד' (6 נק'):

השלימו את הפונקציה הבאה המקבלת שלושה מספרים מסוג int ו-Iterable ומחזירה רשימה **חדשה** המכילה אך ורק את האיברים מסוג Integer שמתחלקים לפחות בשניים מהמספרים.

רמז: השתמשו בסעיפים קודמים.

```
public static List filter(Iterable ds, int d1, int d2, int d3){
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

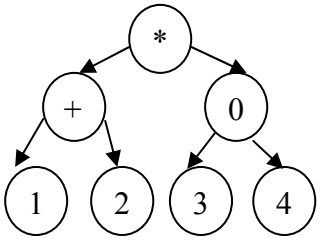
_____

```
}
```

<u>שאלה 2 (20 נק')</u>

<u>הגדרה:</u> עץ בינארי מלא הינו עץ שבו לכל צומת יש 0 או 2 בנים.
ניתן לייצג ביטוי חשבוני באמצעות עץ בינארי מלא שבצמתיו יש אובייקטים מסוג `String`. עץ כזה ייקרא "עץ חשבוני". בעץ חשבוני לכל אופרטור בינארי יש בדיוק זוג סוגריים יחיד, המגדירים את סדר הפעולות באופן יחיד.

לדוגמא:



הביטוי ((1+2)*(3+4)) מיוצג ע"י

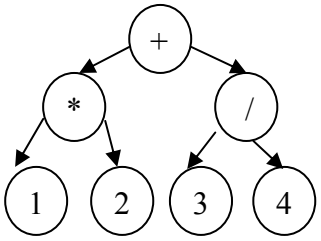
הפעולה בשורש מבוצעת לאחר חישוב הביטויים בתתי העצים שלו.
כל צומת בעץ ממומש ע"י המחלקה BNode:

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

    public BNode(Object data){
         this.data = data;
         left = null;
         right = null;
    }
}
```

<u>סעיף א' (10 נק'):</u>

כתבו שיטה `getExpression(BNode root)` המקבלת שורש של עץ חשבוני המייצג ביטוי חשבוני. השיטה תחזיר מחרוזת המייצגת את אותו ביטוי חשבוני עם סוגריים המגדירים באופן מלא את סדר הפעולות. הניחו שהעץ הינו עץ חשבוני חוקי.
יש להוסיף סוגריים גם אם לא דרוש לסדר הפעולות. לדוגמא, העץ:



מיצג את הביטוי: ((3/4)+(2*1))

(המשך הסעיף בעמוד הבא)

```
public static String getExpression(BNode root){

_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________

}
```

<u>סעיף ב' (10 נק'):</u>

בסעיף זה נשתמש במחלקה StackAsArray המממשת את הממשק Stack בעזרת מערך. הניחו כי הקוד של המחלקה StackAsArray נתון, <u>אין צורך לממש אותו.</u> תזכורת:

```
public interface Stack
{
     public void push(Object o);
     public Object pop();
     public boolean isEmpty();
}
```

<u>רמז:</u> הקוד הבא יאתחל את המחרוזת str למחרוזת שהתוכן שלה הוא ch.

```
char ch = 'a';
String str = "" + ch;
```

ניתן גם להניח כי למחלקה String יש בונה המקבל char כפרמטר ומשתמש בו לאתחל את המחרוזת.

כתבו שיטה המקבלת מחרוזת, המייצגת ביטוי חשבוני עם סוגריים המגדירים באופן מלא את סדר הפעולות, כפי שהוגדר בתחילת השאלה. השיטה תבנה עץ חשבוני המייצג את אותו ביטוי חשבוני ותחזיר את השורש של העץ.

<u>שימו לב:</u> בשאלה זו נניח כי הקלט הנתון הינו מחרוזת ללא רווחים בה המספרים הם שלמים בעלי ספרה אחת בלבד. כמו כן, ניתן להניח כי המחרוזת מבטאת ביטוי חוקי כפי שהוגדרו החוקים של הסוגריים בשאלה ואיננה מחרוזת ריקה או null.

(המשך הסעיף בעמוד הבא)

```
public static BNode getRoot(String exp){

  Stack s = new StackAsArray();

  for (int i=0;i<exp.length();i=i+1)
     if (exp.charAt(i)==')'){
```

_____

_____

_____

_____

_____

_____

_____

_____

```
      }

      else  {
```

_____

_____

_____

_____

```
     }

    return (BNode)s.pop();

}
```

<u>שאלה 3 (20 נק')</u>

נגדיר דמיון בין שתי מחרוזות כמספר האותיות הזהות (בסדרן המקורי) המקסימלי המופיע בשתי המחרוזות.

על מנת לחשב מספר זה, יש "להעמיד" את המחרוזות זו על גבי זו כדי ליצור התאמה, באופן הבא: אותיות יועמדו זו על גבי זו רק במידה והן זהות. לא ניתן לשנות את סדר האותיות במחרוזות, אך ניתן לרווח את המחרוזת כדי ליצור את ההתאמה. הדמיון יתקבל מההעמדה האופטימלית: ההעמדה בה מספר האותיות התואמות הינו רב ביותר.

למשל, הדמיון בין המחרוזות "aacdd" ,"abca" הינו 2, משום שההעמדה אופטימלית הינה:

a	a		c		d	d
a		b	c	a		

תיתכן יותר מהעמדה אופטימלית אחת. למשל, ניתן להעמיד את המחרוזות "aacdd" ,"abca" גם באופנים הבאים:

a	a		c		d	d
	a	b	c	a		

a			a	c	d	d
a	b	c	a			

אלו הן העמדות אופטימליות משום שבכולן יש שתי אותיות ממחרוזת אחת ה"עומדות" על אותיות זהות במחרוזת השניה. מידת הדמיון של שתי המחרוזות הנ"ל הינה 2, משום שלא קיימת העמדה המאפשרת 3 התאמות. במידה ולשתי המחרוזות אין אף אות משותפת, מידת הדמיון ביניהן היא 0.

השלימו את הפונקציה <u>הרקורסיבית</u> alignStrings המקבלת כקלט שתי מחרוזות ומחזירה את מידת הדמיון ביניהן. ניתן להניח שמחרוזות הקלט אינן null.

```
public static  int alignStrings (String s1, String s2) {
      int ans;

      if(_____________________________________________________)

            ans =_____________________________________________;

      else {

            _____________________________________________________

            _____________________________________________________

            _____________________________________________________

            _____________________________________________________

            _____________________________________________________

            _____________________________________________________

            _____________________________________________________

            _____________________________________________________

      }
      return ans;
}
```

<u>שאלה 4 (20 נק')</u>

השאלות הבאות מתייחסות לקטע הקוד המופיע בסוף השאלה (בדף הבא).
בכל סעיף (4 נק') תשובה נכונה אחת. הקיפו את מספר הסעיף בדפי המבחן.

<u>סעיף 1</u>

1) מטרת הפונקציה `insert1` זהה למטרת הפונקציה `insert2` .
2) לאחר ביצוע הפונקציה `insert1`, איברי המערך במקומות `array[0] .. array[i]` יהיו ממוינים, בתנאי שלפני הקריאה לפונקציה, איברי המערך `array[0] .. array[i-1]` היו ממוינים בסדר עולה.
3) לאחר ביצוע הפונקציה `insert1`, איברי המערך במקומות `array[0] .. array[i]` יהיו ממוינים, בתנאי שלפני הקריאה לפונקציה, איברי המערך `array[0] .. array[i-1]` היו ממוינים בסדר יורד.
4) לאחר ביצוע הפונקציה `insert1`, איברי המערך במקומות `array[0] .. array[i]` יהיו ממוינים. המקטע כולו ימויין תוך כדי ביצוע הפונקציה.

<u>סעיף 2</u>

1) רק קריאה לפונקציה what עם מערך שהינו null, תגרור שגיאת קומפילציה.
2) קריאה לפונקציה what עם מערך שהינו null, תגרור שגיאה בזמן ריצה.
3) תתכן שגיאה בזמן ריצה מכיוון שלפונקציות insert1 ו- insert2 חתימות זהות.
4) הקוד הנ"ל שגוי (לא יעבור קומפילציה), משום שלפונקציות insert1 ו- insert2 חתימות זהות.

<u>סעיף 3</u>

1) החלפת שורות 3-6 בשורות הבאות:

```
for (i = 1; i < array.length-1; i = i+2)
      insert2(array, i+1);
```

לא תשנה את ריצת הפונקצית what .
2) בקריאות Math.max(a,b), Math.min(a,b) מעורבים שני מופעים של המחלקה Math.
3) החלפת האופרטור && באופרטור &, בכל אחד משני המקומות בהם מופיע "&&" בפונקציה insert2, לא תשנה את תוצאת ריצת קטע הקוד.
4) לא ניתן לקרוא לפונקציות insert1, insert2 מתוך הפונקציה what, משום שלא ניתן לקרוא לפונקציות שהן private מפונקציה שהינה public.

<u>סעיף 4</u>

1) הפעלת הפונקציה what ממיינת מערך, רק במידה ואורכו זוגי.
2) הפעלת הפונקציה what ממיינת מערך, רק במידה ואורכו אי-זוגי.
3) קיימים מערכים באורך זוגי וכן מערכים באורך אי-זוגי שלא ימוינו ע"י הפעלת הפונקציה what.
4) הפעלת הפונקציה what ממיינת מערך, בסדר עולה.

<u>סעיף 5 (ניתוח זמן ריצה במקרה הגרוע)</u>

1) זמן הריצה של קטע הקוד הנ"ל, על מערך באורך n הינו בסדר גודל של log(n) פעולות.
2) זמן הריצה של קטע הקוד הנ"ל, על מערך באורך n הינו בסדר גודל של n פעולות.
3) זמן הריצה של קטע הקוד הנ"ל, על מערך באורך n הינו בסדר גודל של n*log(n) פעולות.
4) זמן הריצה של קטע הקוד הנ"ל, על מערך באורך n הינו בסדר גודל של n^2 פעולות.

```
1  public static void what(int [] array) {
2     int i;
3     if (array.length>1)
4           insert1(array,1);
5     for (i = 2; i < array.length-1; i = i+2)
6           insert2(array,i+1);
7     if (i < array.length)
8           insert1 (array,i);
9  }

   private static void insert1(int[] array, int i) {
       int value = array[i];
       while (i > 0 && array[i-1] > value){
         array[i] = array[i-1];
         i = i-1;
       }
       array[i] = value;
   }

   private static void insert2(int[] array, int i) {
       int maxValue = Math.max(array[i-1],array[i]);
       int minValue = Math.min(array[i-1],array[i]);

       while (i > 1 && array[i-2] > maxValue){
         array[i] = array[i-2];
         i = i-1;
       }
       array[i] = maxValue;
       i=i-1;
       while (i > 0 && array[i-1] > minValue){
          array[i] = array[i-1];
          i = i-1;
       }
       array[i] = minValue;
   }
```

<u>שאלה 5 (15 נק')</u>

נתונים הממשק והמחלקות:

```
public interface I {}
public abstract class A { private int a = 1; }
public abstract class B extends A { public static int b = 2; }
public class C extends A implements I { private int a=3; }
public class D extends C {}
public class E extends C {}
public class F extends B {}
public class G extends F { public int b=4; }
```

המחלקות D, E ו-F הינן ריקות, כלומר ללא שדות או שיטות. באופן דומה הממשק I הינו ממשק ריק.

כמו כן, נתון קטע הקוד הבא:

```
public static void main(String [] args) {
      C c = new C();
      D d = new D();
      E e = new E();
      F f = new F();
      G g = new G();

// הוספת השורה בכל סעיף הינה במקום זה
}
```

לגבי כל אחת משורות הקוד הבאות (1-15), סמנו מה יקרה אם נכתוב אותה במקום שורת ההערה ב-main – האם תהיה שגיאת קומפילציה, האם התוכנית תיתקל בשגיאה (חריגה) בזמן ריצה, או שהקוד ירוץ בצורה תקינה. במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה (בסעיפים 1-15) **יש לציין את הסיבה**.

<u>שימו לב:</u> הסעיפים הינם בלתי-תלויים.

במקרה של ריצה תקינה של הקוד בסעיפים 11-15 (הכוללים System.out.println) יש לכתוב את הערך שיודפס למסך.

1) e = c;

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

_____

2) c = (C)e;

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

_____

3) e = (E)c;

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

_____

4) d = (D)e;

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

_____

5) c = e;

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

---

6) d = e;

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

---

7) C ci = new I();

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

---

8) A ab = new B();

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

---

9) A af = new G();

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

---

10) I ic = new C();

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה בלבד):

---

11) System.out.println(((G)f).b);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה) או את הפלט (ריצה תקינה):

---

12) System.out.println(c.a);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה) או את הפלט (ריצה תקינה):

---

13) System.out.println(((B)g).b);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה) או את הפלט (ריצה תקינה):

---

14) System.out.println(g.b);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה) או את הפלט (ריצה תקינה):

---

15) System.out.println(B.b);

סמנו מה יקרה: שגיאת קומפילציה / שגיאה בזמן ריצה / ריצה תקינה

ציינו את הסיבה (במקרה של שגיאת קומפילציה או שגיאה בזמן ריצה) או את הפלט (ריצה תקינה):

נספח תזכורת:

```
interface Iterable {
      public Iterator iterator();
}

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

```
public int intValue()
```
שיטה במחלקה Integer המחזירה את הערך

```
int length()
```
שיטה במחלקה String המחזירה את אורך המחרוזת

```
int indexOf(char c)
```
שיטה במחלקה String המחזירה את האינדקס הראשון במחרוזת, בו מופיע התו c,
או 1- אם c לא מופיע כלל במחרוזת

```
char charAt(int index)
```
שיטה במחלקה String המחזירה את התו במיקום index במחרוזת

```
String substring(int beginIndex, int endIndex)
```
שיטה במחלקה String, המחזירה מחרוזת חדשה שהיא תת מחרוזת של עצם המפתח, החל מהמיקום
ה- beginIndex (כולל) ועד המיקום endIndex (לא כולל)

```
double Math.pow(double a, double b)
```
פונקציה המחזירה את הערך של a בחזקת b

```
double Math.sqrt(double a)
```
פונקציה המחזירה את השורש הריבועי החיובי של a

```
double Math.abs(double a)
```
פונקציה המחזירה את ערכו המוחלט של a

```
int Math.abs(int a)
```
פונקציה המחזירה את ערכו המוחלט של a

```
double Math.min(double a, double b)
```
פונקציה המחזירה את הערך המינימלי מבין a, b

```
int Math.min(int a, int b)
```
פונקציה המחזירה את הערך המינימלי מבין a, b

```
double Math.max(double a, double b)
```
פונקציה המחזירה את הערך המקסימלי מבין a , b

```
int Math.max(int a, int b)
```
פונקציה המחזירה את הערך המקסימלי מבין a , b