

שם: \_\_\_\_\_

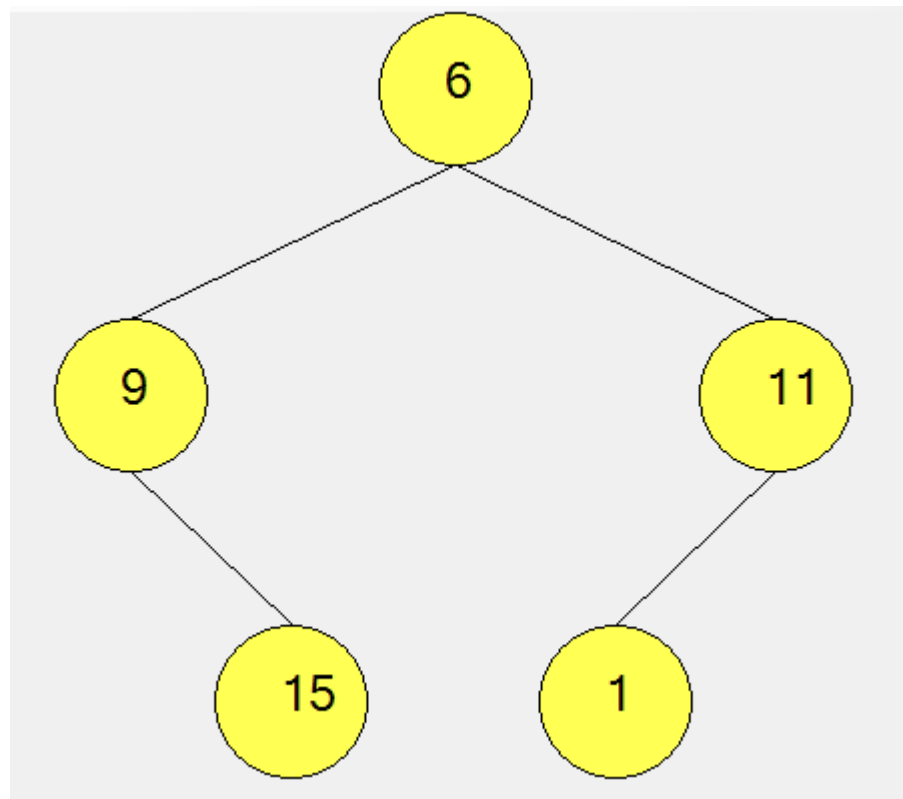
לפניך 4 שאלות, עליך לענות על כל השאלות.  
 שים לב! עליך להעלות את הפתרון לכונן drive הכיתתי. פתרון על דף נייר לא ייבדק.  
 בהצלחה!  
 \*כל שאלה 25 נק'

שאלה 1

"עץ יפה" הוא עץ בינארי המקיים את התנאים הבאים:

- לכל צומת עם שני בנים, ערכו של הבן השמאלי קטן מערכו של הבן הימני
- לכל צומת עם בן יחיד שמאלי, ערך הצומת גדול מערכו של הבן השמאלי.
- לכל צומת עם בן יחיד ימני, ערך הצומת קטן מערכו של הבן הימני.

דוגמה ל"עץ יפה":



**א.** נתונה חתימת הפעולה הבאה:

```
public static boolean isNice(BinNode<Integer> tree);
```

הפעולה מקבלת את הקלט `tree` והוא מצביע לשורש העץ, ומחזירה אמת אם העץ הוא "עץ יפה" (כפי שהוגדר לעיל), אחרת - הפעולה תחזיר שקר. (20 נק')

**ב.** מהי סיבוכיות הפעולה בסעיף א'? נמק בקצרה. (5 נק')

## שאלה 2

נתונה פעולה גנרית count המקבלת מחסנית ומחזירה את מספר האיברים במחסנית. הפעולה שומרת על מבנה המחסנית. סיבוכיות הפעולה היא  $O(n)$  כאשר  $n$  הוא מספר האיברים במחסנית. הנכם רשאים להשתמש בפעולה זו מבלי לממשה.

"מחסנית פלינדרום משולש" היא מחסנית אשר מקיימת את התנאים הבאים:

- מספר האיברים במחסנית מתחלק ב-3.
- האיברים בשליש הראשון והשני יוצרים פלינדרום.
- האיברים בשליש השני והשלישי יוצרים פלינדרום.

דוגמא ל"מחסנית פלינדרום משולש":

`Stack['e','c','f','g','g','f','c','e','e','c','f','g']`

א. כתבו פעולה `isThreePali` המקבלת מחסנית (לא ריקה) מסוג `char` ומחזירה אמת אם היא "מחסנית פלינדרום משולש", אחרת תחזיר שקר. ניתן להסתייע במבני נתונים נוספים מסוג מחסנית בלבד. אין חובה לשמור על מבנה המחסנית. (20 נק')

ב. מהי סיבוכיות הפעולה שכתבתם בסעיף א'? נמק בקצרה. (5 הק')

שאלה 3

א. הסבר את המושגים הבאים ותן דוגמא לכל מושג: (4 נק')

- overloading
- overriding
- Polymorphism

ב. בונים (2 נק'):

הסבר את המושג "בנאי העתקה" (copy-constructor).

ב. הסבר 3 שימושים לפקודה **this**. תן דוגמא לכל שימוש. (3 נק')

ג. נתונות המחלקות הבאות:

```
public class A {
    private int x;
    public A() {
        this.x = 3;
    }
    public A(int x) {
        this.x = x;
        System.out.println(this);
    }
    public int getX() {
        return x;
    }
    public void f() {
        if(x % 2 == 0)
            System.out.println("Even");
        System.out.println("In A's f");
        g();
    }
    public void g() {
        System.out.println("In A's g");
    }
    public String toString() {
        return "X: " + this.x;
    }
}
```

```

public class B extends A{
    private int y;
    public B() {
        super(4);
        y = getX() + 2;
    }
    public B(int y) {
        super();
        this.y = y;
        System.out.println(toString());
    }
    public void h() {
        System.out.println("in B's h");
    }
    public void f() {
        System.out.println("in B's f");
        g();
        h();
    }
    public String toString() {
        return super.toString() + " Y: "+y;
    }
}

```

עבור קטע הקוד הבא:

```
public class Main {  
    public static void main(String[] args) {  
        // ****  
    }  
}
```

לכל אחד מהסעיפים הבאים כתבו:

- האם תתרחש שגיאת הידור (compilation) ואם כן – מהי.
- האם תתרחש שגיאת זמן-ריצה (run-time exception) ואם כן – מהי.
- במידה ואין שגיאות, כתבו מה יהיה הפלט שיודפס על המסך

הנחיות

### שימו לב!

אין כל קשר בין הקוד בסעיפים השונים, לדוגמא –  
ייתכן מצב בו קיימת שגיאה בסעיף כלשהו (x) ובסעיף שלאחר מכן (y) הקוד יהיה תקין.  
לכן עליכם/ן לכתוב עבור סעיף x מהי השגיאה ואילו עבור סעיף y לכתוב מה יהיה הפלט על המסך.

הסעיפים (כל סעיף 3 נק')

1. A a1 = new A();  
a1.f();
2. A a2 = new B(5);  
a2.h();
3. A a3 = new B(2);  
B b1 = (B)a3;  
b1.f();
4. A a4 = new B();  
a4.f();  
System.out.println(a4.toString());
5. B b2 = (B)(new A(6));  
System.out.println(b2.toString());
6. B b3 = (A)(new B());  
System.out.println(b3.toString());

לפניך המחלקות **AA** ו-**BB** :

```

public class AA
{
    private String st;

    public AA() { this.st = "excellent"; }
    public AA(String st) { this.st = st; }
    public String getSt() { return this.st; }
    public void setSt(String st) { this.st = st; }
    public String toString() { return "st = " + this.st; }
}

```

```

public class BB extends AA
{
    private int num;

    public BB() { super(); this.num = 1; }
    public BB(int num , String st) { super(st); this.num = Math.abs(num); }
    public int getNum() { return this.num; }
    public void setNum(int num) { this.num = num; }
    public String toString() { return super.toString() + " num = " + this.num; }
}

```

- א. הגדר במחלקה **AA** פעולה בוליאנית בשם `isLike(Object obj)` המקבלת עצם `obj` מטיפוס `Object`. אם העצם `obj` הינו מטיפוס **AA** וגם תוכן המחרוזת `st` של `obj` זהה לתוכן המחרוזת `st` של העצם הנוכחי — הפעולה תחזיר `true`, אחרת — תחזיר `false`. (5 נק')
- ב. הגדר במחלקה **BB** פעולה הדורסת את הפעולה שהגדרת בסעיף א. אם העצם `obj` הינו מטיפוס **BB** וגם ערך התכונה `num` שלו זהה לערך התכונה `num` של העצם הנוכחי — הפעולה תחזיר `true`, אחרת — תחזיר `false`. (5 נק')

ג. לפניך קטע מפעולה ראשית:

```
AA a = new AA("excellent");  
BB b = new BB();  
a = b;  
if (a.isLike(b)) System.out.println(a);
```

האם קטע התכנית תקין?

אם כן — מה יהיה פלט הקטע? רשום איזו גרסה של הפעולה isLike תופעל — זו של AA או זו של BB.

אם לא — הסבר מהי השגיאה ומתי היא תתגלה: בזמן קומפילציה או בזמן ריצה. (5 נק')

ד. לפניך קטע מפעולה ראשית:

```
AA aa = new AA();  
BB bb = new BB(2, "excellent");  
bb = aa;  
if (bb.isLike(aa)) System.out.println(bb);
```

האם קטע התכנית תקין?

אם כן — מה יהיה פלט הקטע? רשום איזו גרסה של הפעולה isLike תופעל — זו של AA או זו של BB.

אם לא — הסבר מהי השגיאה ומתי היא תתגלה: בזמן קומפילציה או בזמן ריצה. (5 נק')

ה. כתוב פעולה חיצונית בשם longString המקבלת מערך של עצמים מטיפוס Object.

הפעולה מחזירה מחרוזת המורכבת משורשור התכונה st של עצמים מטיפוס AA במערך, באופן הזה:

— אם לעצם יש רק התכונה st, תשורשר המחרוזת שבתכונה st פעם אחת.

— אם לעצם יש גם התכונה num, המחרוזת שבתכונה st תשורשר num פעמים.

— אם אין במערך אף עצם מטיפוס AA, תוחזר מחרוזת ריקה. (5 נק')