נתון מקל באורך n ס"מ.
אפשר לחתוך את המקל לחתיכות בגדלים שונים (בסנטימטרים **שלמים**). לכל גודל של חתיכה יש מחיר. המחירים נתונים במערך בגודל n+1 כך שבתא i נמצא המחיר לחתיכה בגודל i. התא הראשון (אינדקס 0) אינו משמעותי בשאלה זו. הערך (המחיר) בו הוא תמיד 0.

לדוגמא, המערך להלן מכיל את מחירי החתיכות של מקל באורך 8 ס"מ.

0	1	2	3	4	5	6	7	8
0	1	3	10	9	10	17	17	20

עליכם לכתוב שיטה סטטית רקורסיבית המקבלת מערך חד-ממדי, מלא במספרים שלמים חיוביים ממש, ומספר שלם חיובי ממש n המהווה את אורך המקל.
השיטה תחזיר את המחיר המקסימלי שאפשר להשיג על ידי חיתוך המקל ומכירת החתיכות שלו.

בדוגמא לעיל, המחיר המקסימלי שאפשר לקבל הוא 23, כשחותכים את המקל לשלוש חתיכות: אחת באורך 2 ועוד שתי חתיכות באורך 3 (3+10+10 = 23)
אם במערך המחירים היה שינוי, ומחירה של חתיכה באורך 1 היה 4, אז המחיר המקסימלי היה 32, על ידי חיתוך המקל לשמונה חתיכות באורך 1.

שימו לב שמערך המחירים אינו ממוין!

חתימת השיטה היא:

```
public static int findMaxPrice(int [] prices, int n)
```

השיטה צריכה להיות רקורסיבית ללא שימוש בלולאות כלל. גם כל שיטות העזר שתכתבו (אם תכתבו) לא יכולות להכיל לולאות.
אפשר להשתמש בהעמסת-יתר(overloading).
מותר לשנות את המערך במהלך השיטה, אבל בסופה הוא צריך לחזור למצבו המקורי.
אין צורך לדאוג ליעילות השיטה, אבל כמובן שצריך לשים לב לא לעשות קריאות רקורסיביות מיותרות!
אסור להשתמש במשתנים גלובליים (סטטיים)!
אל תשכחו לתעד את מה שכתבתם!

את התשובות יש להקליד בקובץ, לשמור את הקובץ כ- pdf ולהעלות את הקובץ כאן למטה. **לא ייבדקו פתרונות סרוקים או מצולמים.**
ניתן להעלות קבצים מסוג pdf בלבד.

שאלה מספר 1.pdf

שאלה מספר 1:

```
public static int findMaxPrice (int [] prices, int n)
{
        return findMaxPrice(prices, n, 1 );
}

private static int findMaxPrice (int [] prices, int n, int i)
{
        if (i>n)  // חרגנו מהמערך
                return 0;
        if (n==0)  //נגמרו לנו החתיכות
                return 0;
        //המקרה הכללי- או שלוקחים את האיבר וממשיכים או שלא לוקחים אותו וממשיכים
        return Math.max( prices[i]+findMaxPrice(prices, n-i, i) , findMaxPrice(prices, n, i+1) );
}
```

נגדיר:

חציון (median) הוא המספר האמצעי ברשימה מסודרת של מספרים. כך שמספר המספרים הגדולים ממנו ברשימה שווה למספר המספרים הקטנים ממנו ברשימה. כאשר מספר המספרים ברשימה הוא זוגי, נהוג להגדיר את החציון כממוצע בין שני המספרים האמצעיים ברשימה.

כך למשל,
ברשימה {1, 4, 5, 7, 20} החציון הוא 5 - יש ברשימה שני מספרים גדולים ממנו (1 ו- 4) ושני מספרים קטנים ממנו (7 ו- 20).
ברשימה {1, 4, 5, 7, 20, 28} החציון הוא 6 שהוא הממוצע בין 5 ל- 7.

כתבו שיטה המקבלת שני מערכים חד-ממדיים בגודל n מלאים במספרים שלמים, ממוינים בסדר לא יורד. השיטה צריכה להחזיר את החציון של המספרים הנמצאים בשני המערכים.

חתימת השיטה היא:

public static int getMedian (int[]a, int[]b)

אפשר להניח ששני המערכים ממוינים וכן ששניהם באותו אורך. אין צורך לבדוק זאת.

לדוגמא:
אם המערכים a1 ו- a2 הם אלו:

a1 = {1, 12, 15, 26, 38}
a2 = {12, 13, 18, 30, 45}

השיטה תחזיר את הערך 16.

הסבר: אחרי מיזוג שני המערכים למערך ממוין אחד נקבל:
{1, 12, 12, 13, 15, 18, 26, 30, 38, 45}

בגלל שהמערך הממוזג הוא באורך זוגי (2n), החציון הוא הממוצע בין שני המספרים האמצעיים. שני המספרים האמצעיים במערך הם 15 ו- 18, והממוצע ביניהם הוא 16 = 2/ (18 + 15), שזה החציון. (16 ולא 16.5 בגלל חלוקה בשלמים).

שימו לב:
השיטה שתכתבו צריכה להיות יעילה ככל הניתן, גם מבחינת סיבוכיות הזמן וגם מבחינת סיבוכיות המקום. תשובה שאינה יעילה מספיק כלומר, שתהיה בסיבוכיות גדולה יותר מזו הנדרשת לפתרון הבעיה תקבל מעט נקודות בלבד.

כתבו ונמקו מה סיבוכיות זמן הריצה וסיבוכיות המקום של השיטה שכתבתם.
אל תשכחו לתעד את מה שכתבתם!

את התשובות יש להקליד בקובץ, לשמור את הקובץ כ- pdf ולהעלות את הקובץ כאן למטה. לא ייבדקו פתרונות סרוקים או מצולמים.
ניתן להעלות קבצים מסוג pdf בלבד.

שאלה מספר 2:

```
public int getMedin (int[] a, int[] b)
{
        int n= a.length();
        int [] mizugNotSorted= new int [n+n];   // ניצור מערך חדש מאוחד

        for (int i=0; i< n; i++)    // n סיבוכיות = a נכניס למערך החדש הלא ממוין את המערך
                mizugNotSorted[i]= a[i];
        for (int i=0; i< n; i++)    // n סיבוכיות = b נכניס למערך החדש הלא ממוין את המערך
                mizugNotSorted[n+i]= b[i];

        int [] mizug= quickSort (mizugNotSorted); // O(n * log2n). =סיבוכיות
        int hezion=  (mizug[n] + mizug[n+1])/2; // החציון=ממוצע בין שני המספרים האמצעיים
        return hezion;
}
```

- השיטה quicksSort- לקוחה מספר שקפים 2, עמוד 90.
  השיטה בעצם ממיינת מערך
  סיבוכיות זמן הריצה של השיטה הוא O(n * log2n).

- סיבוכיות זמן הריצה שכתבנו הוא O(n * log2n).

n+n+nlogn= n*(1+1+logn)= nlogn

- סיבוכיות המקום של השיטה היא לינארית. o(n)

נניח שהמחלקה Node שלהלן מממשת צומת בעץ בינרי.

```
public class Node
{
    private int _number;
    private Node _leftSon, _rightSon;

    public Node (int number)
     {
            _number = number;
            _leftSon = null;
            _rightSon = null;
    }

    public int  getNumber()       {return _number; }
    public Node getLeftSon()      {return _leftSon; }
    public Node getRightSon()     {return _rightSon; }
}
```

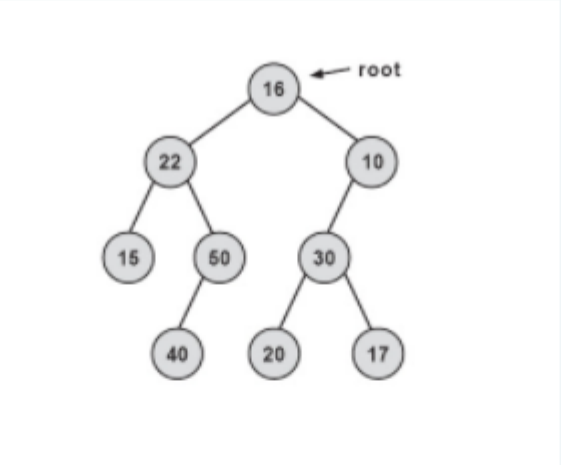
המחלקה BinaryTree מאגדת בתוכה שיטות סטטיות לטיפול בעץ בינרי.
בין השיטות נתונות השיטות f, g ו- secret הבאות:

```
public static int f(Node root)
{
   if (root == null)
      return 0;
   return f(root.getLeftSon()) + 1 + f(root.getRightSon());
}

public static boolean g(Node root)
{
   if (root == null)
      return true;
   if (root.getNumber()%2==1)
      return false;
   return g(root.getLeftSon()) && g(root.getRightSon());
}

public static int secret(Node root)
{
   if (g(root)) {
      return f(root);
   }
   return Math.max(secret(root.getLeftSon()),
                   secret(root.getRightSon()));
}
```

נתון העץ הבינרי הבא, ששורשו הוא root.



ענו על הסעיפים הבאים:

א. איזה ערך תחזיר השיטה **f** כשהיא תקבל כפרמטר את העץ לעיל ששורשו root? התשובה היא (הקלידו מספר בלבד):

9

ב. איזה ערך תחזיר השיטה **g** כשהיא תקבל כפרמטר את העץ לעיל ששורשו root? התשובה היא:

- true
- false

ג. מה יודפס על הפלט לאחר ביצוע הפקודה הבאה:

```
System.out.println (BinaryTree.secret(root));
```

התשובה היא:

2

ד. אם נרצה שההדפסה תהיה 4, עלינו לבצע את **השינוי הבא בעץ** (אם יש כמה שינויים אפשריים, סמנו אחד מהם):

את הערך בצומת

- 17
- 20
- 40
- 30
- 50
- 15
- 10
- 22
- 16

צריך להחליף למספר

- 26
- 55

ה. **אם במקום השינוי של סעיף ד**, נשנה את הערך שבצומת 15 לערך 6. **כמו כן נעשה שינוי נוסף בעץ**: ננתק את התת-עץ ששורשו 20, מאביו 30, ונקבע אותו כבן ימני של הצומת 50, מה יודפס על הפלט לאחר ביצוע הפקודה הבאה:

```
System.out.println (BinaryTree.secret(root));
```

התשובה היא:

5

ו. **אם לאחר השינוי של סעיף ה**, נוסיף לכל אחד מהצמתים את הערך 1. מה יודפס על הפלט לאחר ביצוע הפקודה הבאה:

```
System.out.println (BinaryTree.secret(root));
```

התשובה היא:

1

ז. **אם לאחר השינוי של סעיף ה**, נרצה שההדפסה תהיה 9, עלינו לבצע את השינוי הבא בעץ:

את הערך בצומת

- (●) 17
- ( ) 20
- ( ) 40
- ( ) 30
- ( ) 50
- ( ) 6
- ( ) 10
- ( ) 22
- ( ) 16

צריך להחליף למספר

- (●) 14
- ( ) 25

ח. אם נרצה שהקריאה לשיטה secret תחזיר את הערך 0, אז **בעץ המקורי** מספיק לשנות ערך של צומת אחד בעץ.

- ( ) נכון
- (●) לא נכון

בחבילה מוגדרות חמש מחלקות: A, B, C, D ו- Driver.

להלן מתוארות המחלקות עם החתימות של השיטות שיש בהן. המימוש עצמו אינו חשוב.
שימו לב שלצד כל שיטה כתוב בהערה מספר. זהו מספר השיטה שאנחנו מתייחסים אליו.

```
public class A
{
   public boolean equals(Object obj)  // 1
   public boolean equals(A obj)       // 2
}

public class B extends A
{
   public boolean equals(Object obj)  // 3
   public boolean equals(B obj)       // 4
}

public class C extends A
{
    public boolean equals(A obj)      // 5
}

public class D extends C
{
   public boolean equals(A obj)       // 6
   public boolean equals(B obj)       // 7
   public boolean equals(D obj)       // 8
   public boolean equals(Object obj)  // 9
}

public class Driver
{
    public static void main(String[] args)
    {
          A a1 = new A();
          B b1 = new B();
          C c1 = new C();
          D d1 = new D();
          Object a2 = new A();
          A b2 = new B();
          A c2 = new C();
          C d2 = new D();
          A d3 = new D();
          // כאן יופיעו הפקודות שבשאלות הבאות
    }
}
```

בשאלות הבאות כתובות קריאות לשיטות שונות. הניחו שהן נמצאות בשיטה main שכתובה לעיל.
בכל שאלה עליכם לכתוב אם הקריאה תגרום לשגיאת קומפילציה או לשגיאת ריצה, ואם הקריאה תקינה, עליכם לכתוב איזו שיטה תתבצע בעקבות הקריאה. המספרים מסמנים את מספר השיטה שנקראה, לפי התיאור לעיל.

הקריאה לשיטה:

```
b1.equals(c1);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

הקריאה לשיטה:

```
a1.equals(b2);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

הקריאה לשיטה:

```
b2.equals(b2);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

הקריאה לשיטה:

```
d1.equals(b1);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

הקריאה לשיטה:

```
a1.equals((Object)a1);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

הקריאה לשיטה:

```
a1.equals((A)b2);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

הקריאה לשיטה:

```
a1.equals((B)b2);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

הקריאה לשיטה:

```
b1.equals((C)a1);
```

תגרום לקריאה לאחת מהשיטות equals הממוספרות מ- 1 עד 9, או לשגיאת קומפילציה או לשגיאת הרצה:

יש לבחור תשובה אחת:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- שגיאת קומפילציה
- שגיאה בזמן הרצה

נתונה המחלקה IntNodeSp הבאה, המייצגת איבר ברשימה:

```
public class IntNodeSp
{
   private int _num;
   private IntNodeSp _next, _sp;

   public IntNodeSp(int n) {
      _num = n;
      _next = null;
      _sp = null;
   }

   public IntNodeSp(int num, IntNodeSp n, IntNodeSp nSp) {
      _num = num;
      _next = n;
      _sp = nSp;
   }

   public int getNum()            { return _num;  }
   public IntNodeSp getNext()  { return _next; }
   public IntNodeSp getSp()     { return _sp;   }
   public void setNum (int n)                  { _num = n;    }
   public void setNext (IntNodeSp node)  { _next = node; }
   public void setSp (IntNodeSp node)     { _sp = node;   }
}
```

נתונה רשימה מקושרת של מספרים שלמים ממוינים בסדר לא יורד, הממומשת בעזרת המחלקה IntListSp שלהלן.

בראש הרשימה נמצא המספר הקטן ביותר ובסופה נמצא המספר הגדול ביותר.

איברי הרשימה הם אובייקטים מהמחלקה IntNodeSp שכתובה לעיל.

הבאנו כאן את מימוש הבנאי של המחלקה, ואת מימוש השיטה setSp.

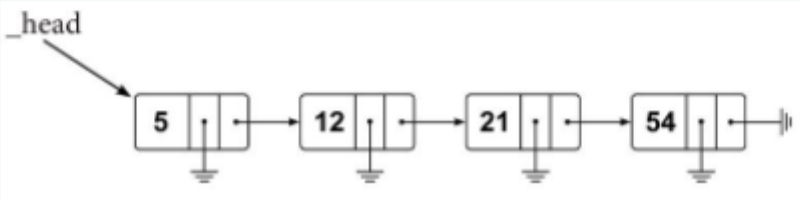
```
public class IntListSp
{
   private IntNodeSp _head;

   public IntListSp( )
   {
      _head = null;
   }
```

```
public void setSp(IntListSp list)
{
    IntNodeSp temp1 = _head;
    while (temp1!=null)
    {
        IntNodeSp temp2 = list._head;
        boolean found = false;
        while ((temp2!=null) && !found)
        {
            if (temp1.getNum()==temp2.getNum())
            {
                if (temp2.getSp()==null)
                {
                    temp2.setSp(temp1);
                    temp1.setSp(temp2);
                    found = true;
                }
                else
                    temp2 = temp2.getNext();
            }
            else
                temp2 = temp2.getNext();
        }
        temp1 = temp1.getNext();
    }
}  //end of setSp
}   //end of class IntListSp
```

אתם יכולים להניח שהרשימה מלאה במספרים שלמים והם ממוינים בסדר עולה.
כמו כן ניתן להניח כי **לפני** הקריאה לשיטה setSp כל המצביעים מסוג sp_ ברשימה עליה הופעלה השיטה וגם בזו המתקבלת כפרמטר הם null.

בשאלות להלן, נסמן את איברי הרשימה כמספרים מופרדים בפסיקים, בתוך סוגריים מסולסלים.
כך לדוגמא, נסמן {5 , 12 , 21 , 54} את הרשימה הזו:



בהנחה שנתונות הרשימות הבאות (בכל רשימה, ראש הרשימה הוא המספר השמאלי):

list1 = {26, 26, 47, 52, 69, 95, 95}

list2 = {15, 26, 26, 69, 69, 95}

לאחר הפעלה של השיטה list1.setSp(list2) ,

המצביע sp_ של 26 השני ברשימה list1 מצביע על:

- 15 ברשימה list2
- 26 השני ברשימה list2
- null
- בחרו בתשובה זו אם אף אחת מהתשובות האחרות אינה נכונה
- 26 הראשון ברשימה list2
- 26 הראשון ברשימה list1
- 47 ברשימה list1
- 69 הראשון ברשימה list2

לאחר הפעלה של השיטה list1.setSp(list2) ,

המצביע sp_ של 69 השני ברשימה list2 מצביע על:

- 52 ברשימה list1
- 69 הראשון ברשימה list2
- 69 ברשימה list1
- 69 השני ברשימה list2
- null
- בחרו בתשובה זו אם אף אחת מהתשובות האחרות אינה נכונה
- 95 הראשון ברשימה list1
- 95 ברשימה list2

מה סיבוכיות הזמן של השיטה setSp במחלקה IntListSp, כשמפעילים אותה על שתי רשימות באורך n?

יש לבחור תשובה אחת:

- $O(\log n)$
- בחרו בתשובה זו אם אף אחת מהתשובות האחרות אינה נכונה
- $O(n)$
- $O(n^3)$
- $O(1)$
- $O(n^2)$

ברצוננו לכתוב את השיטה setSp בצורה יותר יעילה.

השלימו את השיטה:

```
public void setEffSp(IntListSp list)
{
   IntNodeSp temp1 = _head;
   IntNodeSp temp2 = [ list._head ];
   while ((temp1!=null) [ && ] ( [ temp2 != null ] ))
   {
       if ( [ temp1.getNum()==temp2.getNum() ] )
       {
          [ temp2.setSp(temp1) ];
          temp1.setSp(temp2);
          temp1 = [ temp1.getNext() ];
          temp2 = [ temp2.getNext() ];
       }
       else
          if ( [ temp1.getNum()<temp2.getNum() ] )
             temp1 = [ temp1.getNext() ];
          else
             temp2 = [ temp2.getNext() ];
   }
}
}
```

מה סיבוכיות הזמן של השיטה setEffSp במחלקה IntListSp, כשמפעילים אותה על שתי רשימות באורך n?

יש לבחור תשובה אחת:

- $O(n^2)$
- $O(\log n)$
- בחרו בתשובה זו אם אף אחת מהתשובות האחרות אינה נכונה
- $O(n^3)$
- $O(1)$
- $O(n)$