

מזהה קורס: 20441 שם קורס: מבוא למדעי המחשב ושפת Java

מספר שאלה	ציון מירבי	ציון שאלה סופי
1	25.00	25.00
2	25.00	25.00
3.1	2.00	2.00
3.2	3.00	3.00
3.3	3.00	3.00
3.4	3.00	3.00
3.5	6.00	6.00
4.1	1.00	1.00
4.2	2.00	2.00
4.3	2.00	2.00
4.4	2.00	2.00
4.5	2.00	2.00
4.6	2.00	2.00
4.7	2.00	2.00
4.8	2.00	2.00
5.1	2.00	2.00
5.2	5.00	5.00
5.3	3.00	3.00
5.4	8.00	8.00

ציון בחינה סופי : 100.00**הבחינה הבדוקה בעמודים הבאים**

קראו בעיון את ההנחיות שלהלן:

- במבחן יש חמש שאלות. עליכם לענות על **כולן**.
- כל התכניות צריכות להיות מתועדות היטב. יש לכתוב תחילה **בקצרה** את האלגוריתם וכל הסבר נוסף הדרוש להבנת התכנית. יש לבחור בשמות משמעותיים למשתנים, לשיטות ולקבועים שבתכנית. תכנית שלא תתועד כנדרש לעיל תקבל לכל היותר 85% מהניקוד. **אפשר לתעד בעברית**. אין צורך בתיעוד API.
- יש להקפיד לכתוב את התכניות בצורה מבנית ויעילה. **תכנית לא יעילה לא תקבל את מלוא הנקודות**.
- **אם ברצונכם להשתמש בתשובתכם בשיטה או במחלקה הכתובה בחוברת השקפים, אין צורך שתעתיקו את השיטה או את המחלקה למחברת הבחינה**. מספיק להפנות למקום הנכון, ובלבד שההפניה תהיה מדויקת (פרמטרים, מיקום וכו').
- אין להשתמש במחלקות קיימות ב-Java, חוץ מאלו המפורטות בשאלות הבחינה.
- יש לשמור על סדר; תכנית הכתובה בצורה בלתי מסודרת עלולה לגרוע מהציון.
- בכתיבת התכניות יש להשתמש אך ורק במרכיבי השפה שנלמדו בקורס זה. **אין להשתמש במשתנים גלובליים!**

**כל התשובות צריכות להיכתב בתוך קובץ המבחן
במקומות המתאימים בלבד. תשובה שתיכתב שלא
במקומה לא תיבדק.**

שאלה 1 (25 נקודות)

השאלה הבאה מדמה משחק "תפזורת".

נתון מערך דו-ממדי ריבועי המלא בתווים (char). נגדיר שמחרוזת תווים קיימת במערך אם כל תווי המחרוזת נמצאים בסדרם במערך, ואפשר להגיע מתו אחד לתו שאחריו בסדרה על-ידי מעבר מתא לאחד מארבעת שכניו (ימין, שמאל, למעלה, למטה). לא באלכסון.

כך לדוגמא, אם המערך הוא זה:

t	z	x	c	d
s	h	a	z	x
h	w	l	o	m
o	r	n	t	n
a	b	r	i	n

המחרוזת "shalom" קיימת ומסומנת במערך. מתחילה בתא (1,0) ומסתיימת בתא (2,4).

כתבו שיטה **סטטית וקורסיבית** המקבלת מערך שכזה וכן מחרוזת המייצגת מילה. השיטה בודקת אם המילה קיימת במערך או לא. **אם המילה קיימת במערך, עליכם להדפיס את "הנתיב" שלה.** ההדפסה תיעשה בעזרת מטריצת מעקב שתגדירו, שהיא בגודלה של המטריצה המקורית, ובתאים שלה יהיו מספרים שלמים. (בהמשך השאלה יש הסבר מדויק לגבי ההדפסה). בסוף השיטה, במטריצת המעקב צריכים להיות בתאים של המחרוזת, מספרי התווים המרכיבים את המחרוזת, ובתאים שאין בהם תווים של המחרוזת יהיו אפסים. לדוגמא, עבור המערך שלעיל והמחרוזת "shalom" מטריצת המעקב תהיה:

0	0	0	0	0
1	2	3	0	0
0	0	4	5	6
0	0	0	0	0
0	0	0	0	0

אם המילה אינה קיימת יש להדפיס למסך "No Such Word". אם המילה מופיעה יותר מפעם אחת, עליכם להדפיס את אחד ה"נתיבים" בלבד. לא משנה איזה. לא ניתן להשתמש באותו תו יותר מפעם אחת ב"נתיב", כלומר לא ניתן לחזור לתא יותר מפעם אחת. חתימת השיטה היא:

```
public static void findWord(char [][] arr, String word)
```

שימו לב: ניתן להניח שהמערך מכיל אותיות קטנות בלבד וגם במחרוזת יש אותיות קטנות בלבד. אין צורך לבדוק זאת.

לגבי הדפסת הנתוב – עליכם ליצור את מטריצת המעקב, ואז לקרוא לשיטה בשם `printArr` המקבלת כפרמטר מערך דו-מימדי ומדפיסה אותו. **אינכם צריכים לממש את השיטה `printArr` בעצמכם.**

השיטה צריכה להיות רקורסיבית ללא שימוש בלולאות כלל. כך גם כל שיטות העזר שתכתבו (אם תכתבו) לא יכולות להכיל לולאות.

אפשר להשתמש בהעמסת-יתר (overloading).

מוותר 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!

אל תשכחו לתעד את מה שכתבתם!

את התשובה עליכם לכתוב בעמודים הבאים.

```

public static void findWord(char[][] arr, String word)
{
    int[][] output = new int[arr.length][arr.length];
    findWord(arr, word, output, 0, 0, false);
}

public static int findWord(char[][] arr,String word , int[][] output , int i , int j,
boolean done)
{
    if (done == true)
        return 1

    if (j >= arr.length)
        ;return 0

    if (i >= arr.length)
        ;return 1

    if(findWord(arr,word,output,i, j,0))
    }

    ;done = true
    ;printArr(output)
    ;return 1

```

```

{

if (findWord(arr,word,output ,i, j + 1 , false ) == 1)

;return

;return findWord(arr,word,output, i + 1,0, false)

{

private static boolean findWord(char[][] arr, String word, int[][] output, int i, int j,
int stringIndex)
}

if(stringIndex == word.length())

;return true

if(i == arr.length || j == arr.length || i < 0 || j < 0)

;return false

if(arr[i][j] == word.charAt(stringIndex))
}

;output[i][j] = stringIndex + 1

;boolean goLeft, goRight, goUp, goDown

;goLeft = findWord(arr, word, output, i, j-1, stringIndex+1)

;goRight = findWord(arr, word, output, i, j+1, stringIndex+1)

;goUp = findWord(arr, word, output, i-1, j, stringIndex+1)

;goDown = findWord(arr, word, output, i+1, j, stringIndex+1)

```

```
if(goLeft || goDown || goUp || goRight)
```

```
; return true
```

```
else
```

```
}
```

```
; output[i][j] = 0
```

```
; return false
```

```
{
```

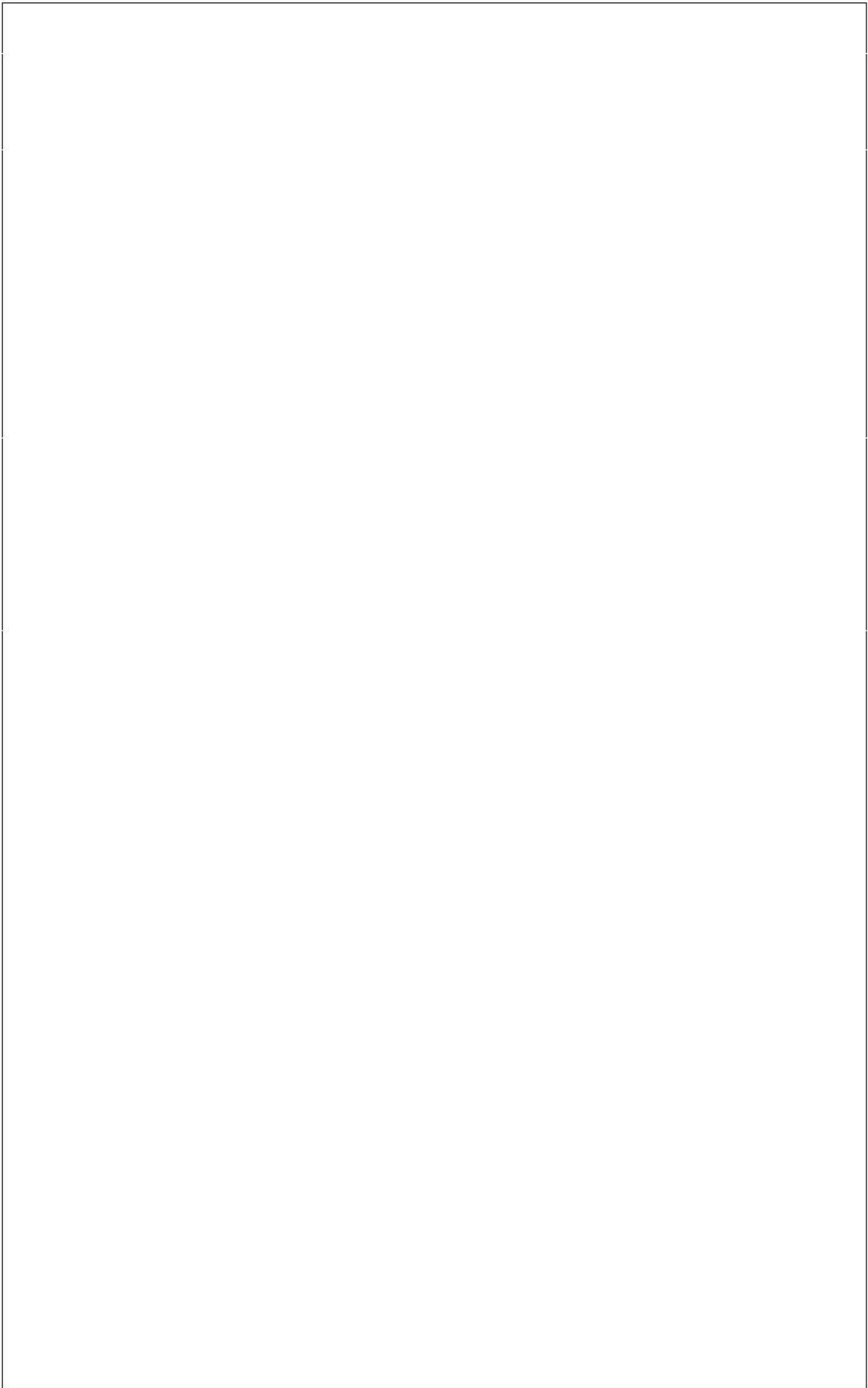
```
{
```

```
; return false
```

```
{
```

25

(1)



שאלה 2 (25 נקודות)

המשפט היסודי של האריתמטיקה אומר כי אפשר לפרק כל מספר טבעי (כלומר שלם חיובי) לגורמים ראשוניים, והפירוק הזה הוא יחיד. (סדר הגורמים אינו משנה) למשל, את המספר 40 ניתן לפרק לגורמים ראשוניים הבאים:

$$2 * 2 * 2 * 5 = 2^3 * 5^1 = 40.$$

המספרים 2 ו-5 הם מספרים ראשוניים. (כזכור, מספר ראשוני הוא מספר שמתחלק רק ב-1 ובעצמו ללא שארית).

שימו לב שאין שום דרך אחרת לכתוב את המספר הזה בתור מכפלת ראשוניים. לכן הפירוק הזה הוא יחיד.

חלק מהמספרים הם מכפלה של **שני מספרים ראשוניים בלבד**. לא כמו המספר 40 לעיל, שהגורם הראשוני שלו 2, מופיע 3 פעמים במכפלה. **לדוגמא,**

- המספר 35 הוא מכפלה של 5 ו-7.
- המספר 493 הוא מכפלה של 17 ו-29.
- המספר 8509 הוא מכפלה של 67 ו-127.

הניחו כי **נתונה לכם** השיטה הבוליאנית הבאה: **אין צורך לממש את השיטה הזו.**

```
public boolean isHighFactorInRange(int n, int low, int high)
```

שיטה זאת מקבלת מספר טבעי n ו-2 מספרים low ו- $high$ - אשר מייצגים שני קצוות של טווח חיפוש. **אנו מניחים, כי n הוא מכפלה של בדיוק שני מספרים ראשוניים p ו- q .** השיטה בודקת האם הגורם הראשוני הגדול מבין p, q נמצא בטווח החיפוש. אם כן היא מחזירה `true`, אחרת היא מחזירה `false` (טווח החיפוש כולל את הקצוות).

לדוגמא,

אנחנו יודעים כי המספר 8509 הוא מכפלה של 67 ו-127. לכן, אם נקרא לשיטה isHighFactorInRange עם המספר 8509, $n = 8509$, וטווח החיפוש $100 - 1$ ($low = 1, high = 100$), השיטה תחזיר false. כיון שהגורם הראשוני הגדול יותר מבין 67 ו-127 הוא 127, והוא לא נמצא בטווח בין 1 ל-100. אם היינו נותנים טווח $low = 1, high = 200$, השיטה היתה מחזירה true, כי 127 הוא בין 1 ל-200.

שימו לב שהפרמטרים לשיטה לא כוללים את הגורמים p ו- q עצמם. רק את המספר n ואת טווח החיפוש. זכרו ש- n הוא מאותם מספרים שהם מכפלה של בדיוק שני מספרים ראשוניים p ו- q .

עליכם לכתוב שיטה בשם findFactors שתקבל מספר n שהוא שלם חיובי והוא מכפלה של 2 מספרים ראשוניים, p ו- q , $(q * p = n)$. השיטה תמצא ותדפיס את p ו- q .

חתימת השיטה:

```
public void findFactors (int n)
```

לדוגמה, עבור $n=8509$ השיטה תדפיס 67 ו-127 מאחר ו- $67*127=8509$.

כמובן שכדאי לכם מאד להשתמש בשיטה isHighFactorInRange הנתונה לכם.

להזכירכם, אינכם צריכים לממש את השיטה isHighFactorInRange, אלא רק להשתמש בה.

אין צורך להתייחס לסיבוכיות של השיטה isHighFactorInRange, אתם יכולים להניח שהשיטה יעילה. זה לא משנה מהי הסיבוכיות שלה. אתם צריכים להתייחס רק לסיבוכיות של השיטה findFactors שאתם כותבים.

שימו לב:

השיטה findFactors 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.

מה סיבוכיות זמן הריצה וסיבוכיות המקום של השיטה שכתבתם? הסבירו תשובתכם.

אל תשכחו לתעד את מה שכתבתם!

את התשובה עליכם לכתוב בעמודים הבאים:

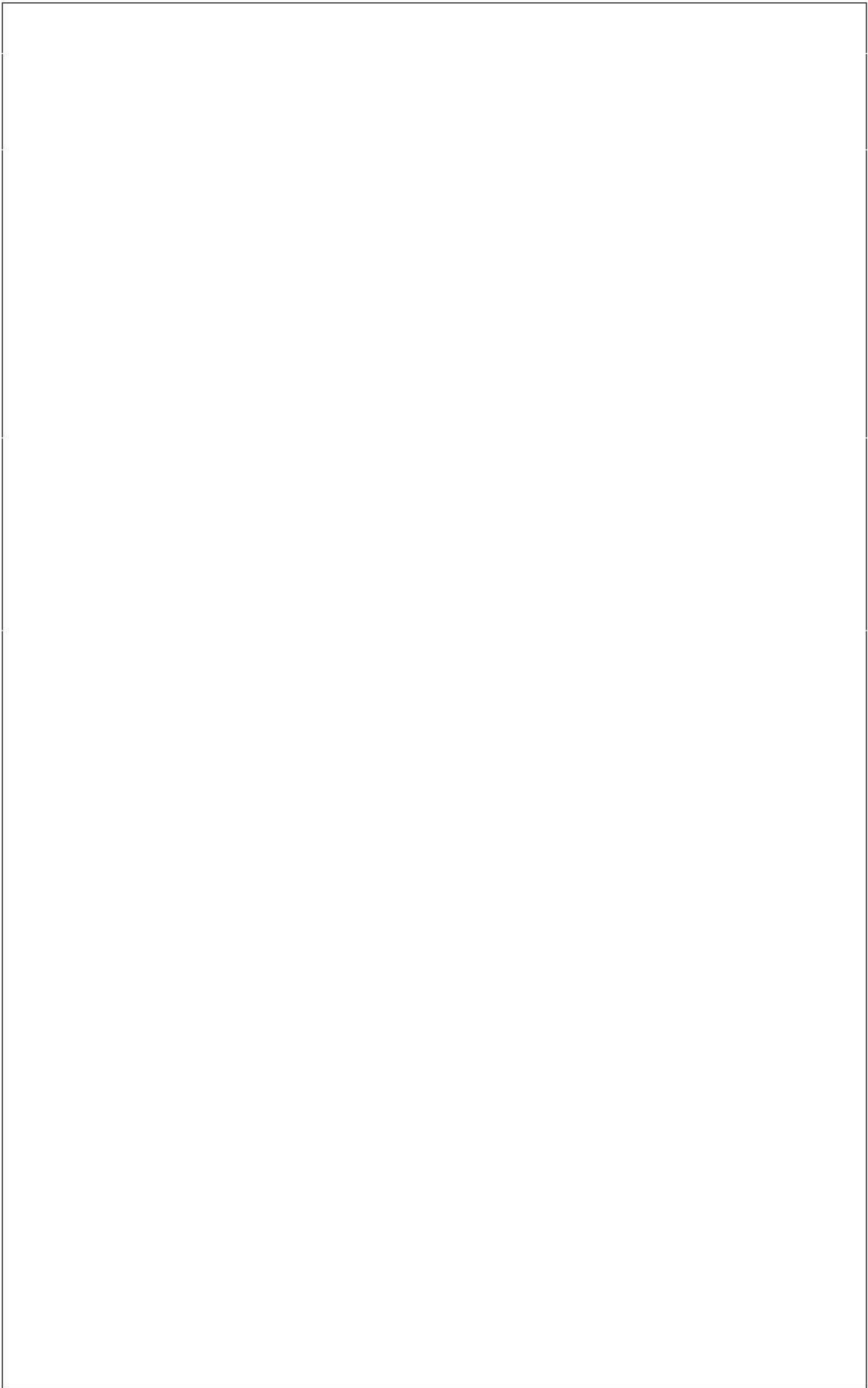
```

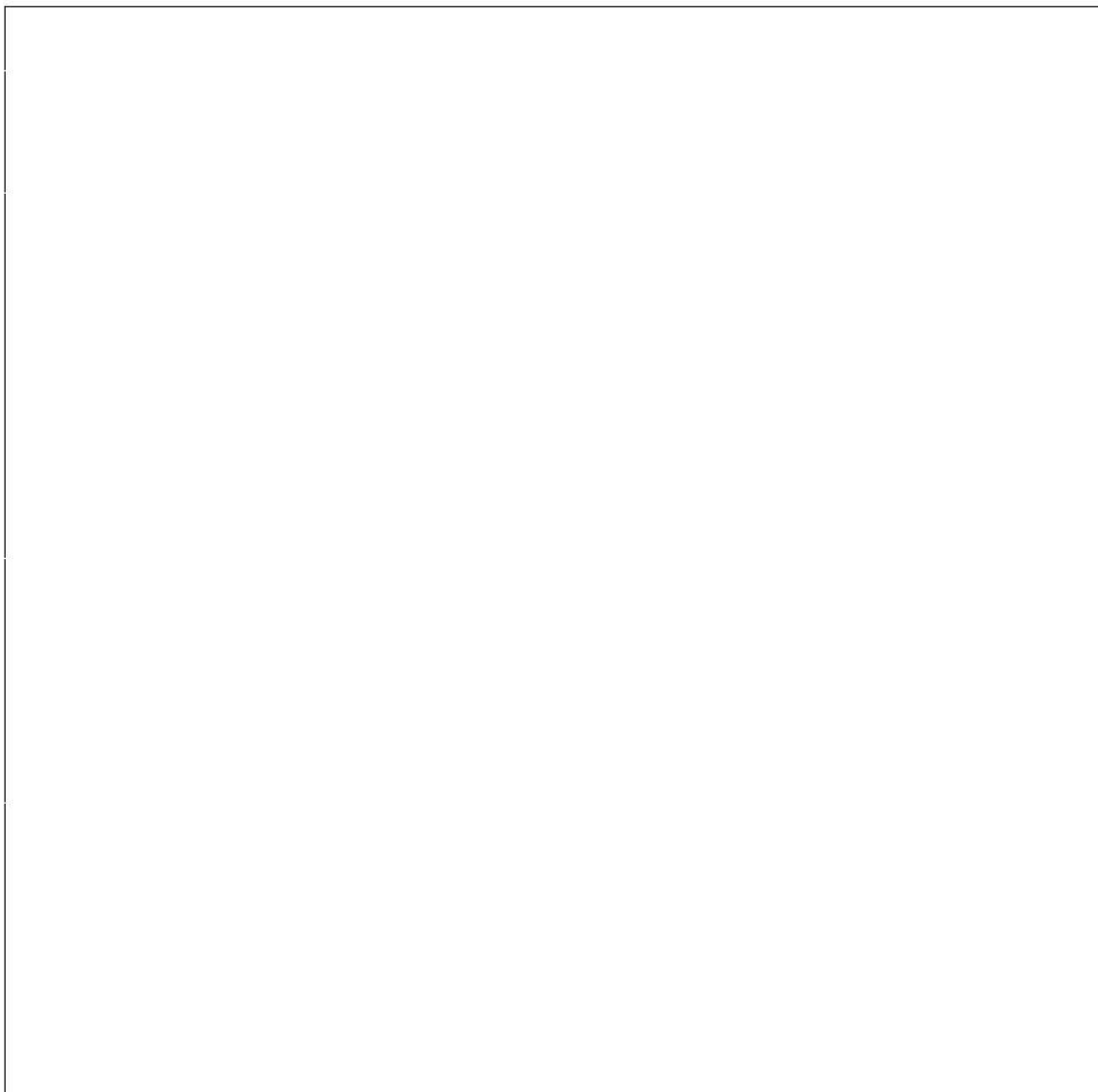
public static void findFactors(int n) // time complexity: O(log n)
{
    // space complexity: O(1)

    int low = 2, high = n; // low = smallest prime, high = request number
    while(low <= high)
    {
        int mid = (low+high)/2; // finds the middle point
        boolean leftHalf = isHighFactorInRange(n, low, mid); // checks range low-mid
        boolean rightHalf = isHighFactorInRange(n, mid, high); // checks mid-high
        if(leftHalf && rightHalf) // only happens if mid is biggest factor
        {
            System.out.println(mid + " " + n/mid);
            break;
        }
        if(leftHalf) // biggest factor is in left half
        {
            high = mid - 1; // changes search range, decreasing the maximum value
            continue; // and continues
        }
        else
        {
            low = mid + 1; // changes search range, increasing the minimum value
            continue; // and continues
        }
    } //end of while
    return; // generic return for a void method
} // end of method... indentation can be better, but the lines are too long to indent properly.

```

25
(2)





שאלה 3 (17 נקודות)

נניח שהמחלקה Node שלהלן מממשת צומת עץ בינרי שערכיו הם מספרים שלמים (int).
שימו לב שבנוסף לשני המצביעים לבן השמאלי ולבן הימני, יש לכל אובייקט במחלקה Node גם שדה father שמצביע על האב של הצומת.
בשורש העץ, המצביע לאב הוא null.

Constructor Summary

[Node](#)(int num)

Constructs a Node object.

Method Summary

<u>Node</u>	<u>getFather()</u> Returns the father of the node.
<u>Node</u>	<u>getLeftSon()</u> Returns the left son of the node.
int	<u>getData()</u> Returns the value of the node.
<u>Node</u>	<u>getRightSon()</u> Returns the right son of the node.

המחלקה BinaryTree מאגדת בתוכה שיטות סטטיות לטיפול **בעץ חיפוש בינרי**.
נתונות **חמש** השיטות הבאות: f, g, what, secret ו-something המקבלות כפרמטר שורש של עץ
חיפוש בינרי (מטיפוס Node):

```
public static Node f(Node root)
{
    if (root.getRightSon() == null)
        return root;
    return f (root.getRightSon());
}
```

```

private static Node g (Node node)
{
    if (node.getFather() == null)
        return null;
    if (node.getFather().getRightSon() == node)
        return node.getFather();
    return g(node.getFather());
}

public static Node what (Node root)
{
    if (root == null)
        return null;
    if (root.getLeftSon() != null)
        return f (root.getLeftSon());
    return g (root);
}

public static boolean secret(Node root)
{
    if (root == null)
        return true;

    Node temp1 = what(root);
    if (temp1 == null)
        return true;

    Node temp2 = what(temp1);
    if (temp2 == null)
        return true;

    if (root.getData() != temp1.getData() + temp2.getData())
    {
        System.out.println(root.getData() + " != " +
            temp1.getData() + " + " + temp2.getData());
        return false;
    }


    return secret(temp1);
}

public static boolean something(Node root)
{
    return secret (f(root));
}

```

סעיף א (2 נקודות):


מה מבצעת השיטה f כאשר היא מקבלת שורש של **עץ חיפוש בינרי** כלשהו? הסבירו בקצרה **מה** מבצעת השיטה ולא **כיצד** היא מבצעת זאת. מה משמעותו של הערך שהוחזר? **התייחסו למקרי קצה** **התשובה היא:**



בעץ חיפוש בינארי הערך שיוחזר הוא הגדול ביותר שבעץ, והוא נמצא כהצאצא הכי ימיני.
במידה ואין צומת ימיני ל root יוחזר root עצמו.

סעיף ב (3 נקודות):

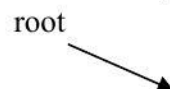
מה מבצעת השיטה **what** כאשר היא מקבלת **צומת כלשהו בעץ חיפוש בינרי** כלשהו? הסבירו בקצרה **מה** מבצעת השיטה ולא **כיצד** היא מבצעת זאת. מה משמעותו של הערך שהוחזר? שימו לב שהפרמטר המועבר אינו בהכרח שורש העץ. **התייחסו למקרי קצה**. אפשר להניח שהצומת נמצא בעץ.
התשובה היא:

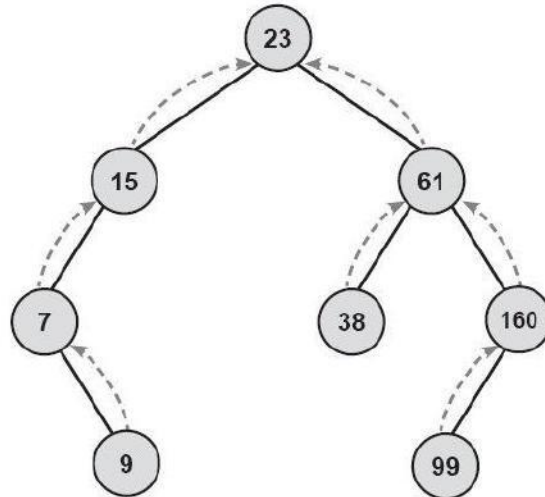


אם השיטה מקבלת צומת תקין, מוצאת את המספר הגדול ביותר שעדיין קטן מן ערך הצומת ההתקבלה כפרמטר. אם אין כזה, מוצאת את המספר הכי קטן שהוא עדיין יותר גדול מן ערך הצומת ההתקבלה כפרמטר.
אחרת מחזירה null

סעיף ג (3 נקודות):

בהינתן עץ חיפוש בינרי ששורשו root המצויר להלן:





איזה ערך תחזיר השיטה **something** ומה היא תדפיס על הפלט, אם נקרא לה עם הפרמטר root שהוא שורשו של העץ לעיל?

התשובה היא:



$23 \neq 9 + 15$

false

סעיף ד (3 נקודות):

איזה שינוי **מינימלי** אפשר לעשות **על העץ** לעיל, כדי שהקריאה לשיטה (root) **something** תחזיר ערך אחר מזה שהוחזר בסעיף ג? **שימו לב, השינוי צריך להיות בעץ ולא בשיטה (כגון, הוספת צומת, הורדת צומת, שינוי ערך של צומת וכד')**. מינימלי במובן של מינימום פעולות על העץ. כך, אם מורידים צומת, ויש לצומת הזה בנים, במספר הפעולות נספרים גם הבנים של הצומת שהורדו.

התשובה היא:



נשנה את ערך הצומת עם הערך 9 לערך 8.

סעיף ה (6 נקודות):

מה מבצעת השיטה **something** באופן כללי כשהיא מקבלת כפרמטר שורש של עץ **חיפוש בינרי** root? שימו לב, עליכם לתת תיאור ממצה של מה עושה השיטה באופן כללי, ולא תיאור של מה עושה כל שורה בשיטה, או איך היא מבצעת זאת. **כלומר, מה המשמעות של הערך שהשיטה מחזירה? התייחסו למקרי קצה.**

התשובה היא:

השיטה בודקת אם עבור כל מספר החל מן האיבר הכי גדול, סכום של שני המספרים הגדולים ביותר שעדיין קטנים ממנו שווה לאיבר עצמו. אם כן, חוזרת על התהליך עם המספר השני הכי גדול.

תחזיר השיטה true אם הבדיקה נכונה לכל המספרים, או FALSE עם שני המספרים ששברו את השיוויון הדרוש.



שאלה 4 (15 נקודות)

נתונות המחלקות One, Two, Three, Four, Five הבאות, כל אחת בקובץ נפרד, כמובן, וכולן באותן חבילה:

```
public class One
{
    protected int _a;
    public One() {
        _a=1;
    }
    public One(int a) {
        _a=a;
    }
    public int getA(){
        return _a;
    }
    public int f(){
        return _a;
    }
}
//-----

public class Two extends One
{
    public Two(){
        super();
    }
    public Two(int a){
        super(a);
    }
    public int f(){
        return _a+1;
    }
}
```

```
    }  
}  
//-----  
  
public class Three extends Two  
{  
    public Three(){  
        super();  
    }  
    public Three(int a){  
        super(a);  
    }  
    public int f(){  
        return _a+2;  
    }  
    public int g(){  
        return _a;  
    }  
}
```

```

public class Four extends Three
{
    public Four(){
        super();
        _a++;
    }
    public Four(int a){
        super(a);
    }
    public Four(int a,int b){
        super();
        _a=_a+a+b;
    }
    public int f(){
        return _a-1;
    }
}
//-----

public class Five extends Three
{
    public Five(){
        super();
    }
    public int g()
    {
        return _a+1;
    }
}

```

כתבנו שיטה המקבלת אובייקט של אחת המחלקות One, Two, Three, Four, Five לעיל, ומדפיסה מהו סוג האובייקט. כלומר, מאיזו מחלקה הוא נוצר. עליכם להשלים את החסר בשיטה.

לדוגמא,

אם היינו כותבים ב- main את הפקודה:

```
Three t = new Three(5);
```

```
whichClass(t)
```

ואחר כך היינו קוראים לשיטה

אז היה מודפס על הפלט:

```
Three
```

חתימת השיטה היא:

```
public static void whichClass(Object obj)
```

שימו לב, עליכם להשלים את השיטה שאנחנו כתבנו. אם לא תעשו כך, ותכתבו שיטה משלכם, עליכם לעמוד בהגבלות הכתובות להלן, ובכל מקרה, התשובה שלכם תקבל לכל היותר 10 נקודות, גם אם היא תהיה נכונה.

כדי לבדוק מהו סוג האובייקט, יש להיעזר בשיטות `getA`, `f`, `g` שהוגדרו במחלקות לעיל. אסור להשתמש בפעולה `instanceOf` או בפעולות אחרות של המחלקה `Object`. אסור לשנות את המחלקות `One`, `Two`, `Three`, `Four`, `Five`. אפשר להניח שהאובייקט `obj` שייך לאחת המחלקות הנ"ל, והוא אינו `null`.

התשובה היא:

```

(4.1) public static void whichClass(Object obj)
{
    (4.2) int a = ((One) obj).getA();
    int f = ((One) obj).f();
    int g;
    (4.3) if(a==f)
        System.out.println("One");
    (4.4) if(a+1 == f)
        System.out.println("Two");
    (4.5) if(a - 1 == f)
        System.out.println("Four");
    if(a==f-2)

    {
        (4.6) g= ((Three) obj).g();
        (4.7) if (g == a)
            (4.8) System.out.println("Three");
        else
            System.out.println("Five");
    }
}
    
```

שאלה 5 (18 נקודות)

נגדיר: "תור ישראלי" (Israeli Queue) הוא מעין תור רגיל (Queue) שהכניסה אליו נעשית מסוף התור (זנבו), והיציאה מהתור היא מראשו, אלא שבתור ישראלי ניתן גם להידחף ישירות לראש התור. היציאה מהתור הישראלי היא תמיד מראשו.

המחלקה IsraeliQueue מממשת תור ישראלי שהאיברים שבו הם מספרים שלמים.

להלן נתונות השיטות במחלקה IsraeliQueue הפועלות על תור ישראלי:

IsraeliQueue()	בנאי הבונה תור ישראלי ריק
void insertAtEnd (int t)	השיטה מכניסה את t לסוף התור הישראלי
void insertAtHead (int t)	השיטה מכניסה את t לראש התור הישראלי
int removeFromHead()	השיטה מוציאה את האיבר מראש התור הישראלי ומחזירה אותו
boolean isEmpty()	השיטה מחזירה "אמת" אם התור הישראלי ריק ו-"שקר" אחרת

לפניכם שתי שיטות המוגדרות בתוך המחלקה IsraeliQueue:

```
public static int what(int [] a, int x, int y)
{
    int t = a[x];
    IsraeliQueue q = new IsraeliQueue();
    q.insertAtEnd(t);
    int p=x;
    for (int i = x+1; i <= y; i++) {
        if (a[i]<t) {
            q.insertAtHead(a[i]);
            p++;
        }
        else
            q.insertAtEnd(a[i]);
    }
    for (int i = x; i <= y; i++)
        a[i] = q.removeFromHead ();
    return p;
}
```

```
public static void something(int [] a)
{
    int p = a.length;
    while (p!=0)
        p = what(a, 0, p-1);
}
```

נתון המערך **a** הבא:

0	1	2	3	4	5	6	7	8	9
7	5	8	2	12	1	2	6	9	3

סעיף א (2 נקודות)

איזה ערך תחזיר השיטה **what** לאחר הקריאה **what(a, 0, 9)**? כיצד יראה המערך לאחר הקריאה הזו?

התשובה היא:

{3, 6, 2, 1, 2, 5, 7, 8, 12, 9}

שיטה תחזיר 6



סעיף ב (5 נקודות)

מה מבצעת השיטה **what** כאשר היא מקבלת **מערך חד-ממדי** כלשהו מלא במספרים שלמים, ושני פרמטרים שלמים x ו- y שמייצגים את האינדקס של התא הראשון והאינדקס של התא האחרון במערך (בהתאמה)? הסבירו בקצרה **מה** מבצעת השיטה ולא **כיצד** היא מבצעת זאת. מה משמעותו של הערך שהוחזר? **התייחסו למקרי קצה**.

התשובה היא:

משאירה את כל האיברים שבאינדקס פחות מן X במקומם, ואז משנה את סדר הערכים בתת-מערך מן מקום X עד Y , כך שכל האיברים שערכם גדול מן $a[X]$ נמצאים מאחוריו, וכל האיברים שיותר קטנים מן $a[X]$ הם לפניו.

השיטה מחזירה מספר האיברים שקטנים מן $a[x]$ בתת-מערך שצדדיו x, y

במקרה $X > Y$ או $a.length > X$ ניתקל בשגיאת ריצה. במקרה $A = null$ נקבל שגיאת ריצה (גישה ל NULL)



סעיף ג (3 נקודות)

כיצד יראה המערך a לאחר הקריאה something(a)?

התשובה היא:

{1, 2, 2, 3, 6, 5, 7, 8, 12, 9}



סעיף ד (8 נקודות)

מה מבצעת השיטה something כאשר היא מקבלת מערך חד-ממדי כלשהו מלא במספרים שלמים? הסבירו בקצרה מה מבצעת השיטה ולא כיצד היא מבצעת זאת. התייחסו למקרי קצה.

התשובה היא:

מסדרת את המערך כך שהאיבר המינימום של המערך נמצא בראש התור.



בהצלחה!