האוניברסיטה הפתוחה

כ"א באדר ב' תשפ"ב

24

במרץ 2022

מס' שאלון - 476

סמסטר 2022א

מס' מועד 74

4 / 20454

שאלון בחינת גמר

20454 - מבוא למדעי המחשב ושפת Java ב

משך בחינה: 3 שעות

בשאלון זה **15** עמודים

מבנה הבחינה:

קראו בעיון את ההנחיות שלהלן:
* בבחינה יש חמש שאלות.
* כל התכניות צריכות להיות מתועדות היטב.
יש לכתוב תחילה בקצרה את האלגוריתם וכל הסבר נוסף הדרוש להבנת התכנית.
יש לבחור בשמות משמעותיים למשתנים, לפונקציות ולקבועים שבתכנית.
תכנית שלא תתועד כנדרש לעיל תקבל לכל היותר % 85 מהניקוד.
* 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!
* את התשובות לשאלות 3 - 5 יש לכתוב על גבי השאלון. לא נבדוק תשובות שייכתבו במקום אחר!
* אפשר לתעד בעברית. אין צורך בתיעוד API .

חומר עזר:
חוברות השקפים 7-12.
אין להכניס חומר מודפס נוסף או חומר אחר מכל סוג.
אין להכניס מחשב או מחשבון או מכשיר אלקטרוני מכל סוג שהוא.

בהצלחה !!!

חלק א – עליכם לענות על כל השאלות בחלק זה במחברת הבחינה

שאלה 1 (25 נקודות)

נתון מערך דו-ממדי mat, בגודל n×m (n שורות ו- m עמודות) שערכיו הם תווים (char).

כמו כן נתונה מחרוזת תווים pattern.

נגדיר **מסלול-אותיות** (letter-path) במערך אם הוא מקיים את התנאים הבאים:

1. המסלול מתחיל בתא הראשון במערך – שורה ראשונה ועמודה ראשונה.
2. אפשר לעבור מתא אחד לשני רק אם הוא שכן שלו מימין, משמאל, למעלה או למטה. לא באלכסון.
3. אפשר ללכת מתא לתא רק אם גם התא הבא מכיל תו שהוא אחד מהתווים במחרוזת pattern.
4. **המסלול לא יכול לחזור לתא בו הוא היה כבר!**

סעיף א: (15 נקודות)

כתבו שיטה סטטית רקורסיבית בשם lengthPath המקבלת מערך דו-ממדי mat מלא בתווים, ומחרוזת תווים pattern. השיטה צריכה להחזיר את אורכו של מסלול האותיות הארוך ביותר שיש במערך.

אם בתא הראשון במערך – שורה ראשונה ועמודה ראשונה - מופיע תו שלא נמצא ב-pattern, השיטה צריכה להחזיר 0.

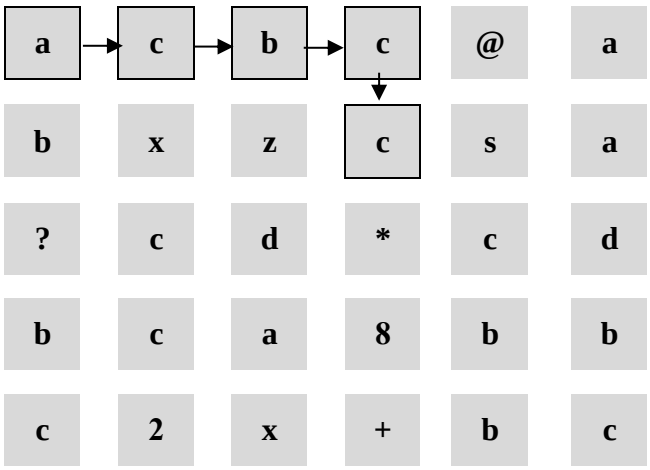
חתימת השיטה היא:

```
public static int lengthPath (char[][] mat, String pattern)
```

לדוגמא, נניח שהמחרוזת pattern היא "abc"

במערך mat להלן מסומן מסלול-אותיות באורך מקסימלי – **5**.

שימו לב שבמערך יש עוד מסלול-אותיות באורך 2 a-b, אבל הוא קצר יותר מהמסלול המסומן.



```java
public static int lengthPath (char[][] mat, String pattern) {
    return lengthPath(mat,pattern,0,0,0,0,1,0);
}
public static int lengthPath(char[][] mat, String pattern, int i, int j, int prevI,int privJ, int pathLength, int charIndex) {

    int maxPath = pathLength;
    char currentChar = pattern.charAt(charIndex);

    if (j > 0 && (j-1 != privJ || i != prevI) && mat[j-1][i] == currentChar) {
        int path = lengthPath(mat, pattern, i, j-1, i, j, pathLength+1, 0);
        maxPath = Math.max(maxPath, path);
    }
    if (i > 0 && (j != privJ && i-1 != prevI) && mat[j][i-1] == currentChar) {
        int path = lengthPath(mat, pattern, i-1, j, i, j, pathLength+1, 0);
        maxPath = Math.max(maxPath, path);
    }
    if (i < mat.length && (j+1 != privJ || i != prevI) && mat[j+1][i] == currentChar) {
        int path = lengthPath(mat, pattern, i, j+1, i, j, pathLength+1, 0);
        maxPath = Math.max(maxPath, path);
    }
    if (j < mat[j].length && (j != privJ || i+1 != prevI) && mat[j][i+1] == currentChar) {
        int path = lengthPath(mat, pattern, i+1, j, i, j, pathLength+1, 0);
        maxPath = Math.max(maxPath, path);
    }

    if (charIndex < pattern.length()-1) {
        int path = lengthPath(mat, pattern, i, j, prevI, privJ, pathLength,charIndex+1);
        maxPath = Math.max(maxPath, path);
    }
    return maxPath;
}
```

סעיף ב: (10 נקודות)

נגדיר שמסלול האותיות יכול להתחיל בכל תא במערך, ולא דווקא בתא הראשון.

כתבו שיטה סטטית רקורסיבית בשם maxPath המקבלת מערך דו-ממדי mat מלא בתווים, ומחרוזת תווים pattern. השיטה צריכה להחזיר את אורכו של מסלול האותיות הארוך ביותר שיש במערך.

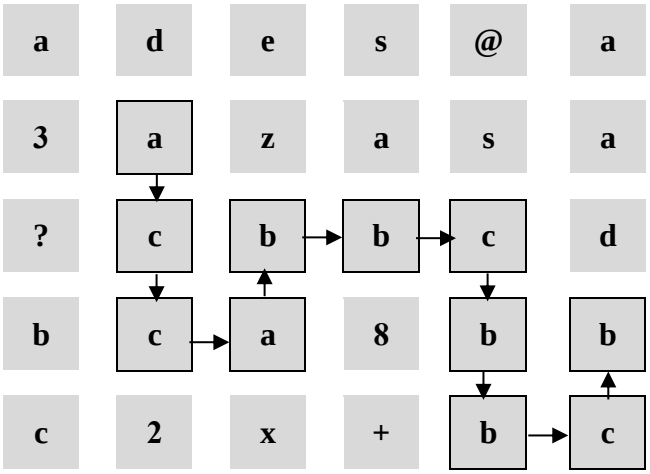
אם באף תא במערך לא מופיע תו שנמצא ב-pattern, השיטה צריכה להחזיר 0.

לדוגמא, נניח שהמחרוזת pattern היא "abc"

במערך mat להלן מסומן מסלול-אותיות באורך מקסימלי – **11**.

שימו לב שבמערך יש עוד 2 מסלולי-אותיות באורך מקסימלי 1. (שניהם מתחילים בתא שבשורה האחרונה ובעמודה הראשונה).

1. c-b-c-a-b-b-c-b-b-c-b
2. c-b-c-c-b-b-c-b-b-c-b



חתימת השיטה היא:

```
public static int maxPath (char[][] mat, String pattern)
```

אתם יכולים להשתמש בשיטה lengthPath מסעיף א גם אם לא כתבתם אותה.

שימו לב שבמקרה כזה אפשר להניח שחתימת השיטה מסעיף א היא זו שכתובה למעלה בסעיף א, או:

```
public static int lengthPath (char[][] mat, String pattern, int row, int col)
```

כאשר row ו- col הם האינדקסים של השורה והעמודה המתאימים.

הנחיות לשני הסעיפים:

- השיטות שתכתבו צריכות להיות רקורסיביות ללא שימוש בלולאות כלל. כך גם כל שיטות העזר שתכתבו (אם תכתבו) לא יכולות להכיל לולאות.
- אפשר להשתמש בהעמסת-יתר (overloading).
- אפשר להניח שהמטריצה אינה null ואינה ריקה. כמו כן אפשר להניח שמספר העמודות בכל השורות שווה (כלומר, המטריצה היא מלבנית). אין צורך לבדוק זאת!
- אפשר להניח שהמחרוזת אינה null ואינה ריקה.
- 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!
- כדי לדעת אם תו נמצא במחרוזת, אפשר להשתמש בשיטה indexOf של המחלקה String, המחזירה את האינדקס של התו במחרוזת, ואם הוא לא נמצא, מחזירה -1.
- **אל תשכחו לתעד את מה שכתבתם**.

```java
public static int maxPath(char[][] mat, String pattern) {
    return maxPath(mat, pattern, 0, 0, 0, 0);
}

private static int maxPath(char[][] mat, String pattern, int xIndex, int yIndex, int charIndex, int sum) {

    int pathSum = sum;
    char currentChar = pattern.charAt(charIndex);
    char element = mat[yIndex][xIndex];

    if (currentChar == element) {
        int path = lengthPath(mat, pattern);
        pathSum = Math.max(pathSum, path);
    }
    if (charIndex < pattern.length() - 1) {
        int path = maxPath(mat, pattern, xIndex, yIndex, charIndex + 1, sum);
        pathSum = Math.max(pathSum, path);
    } else {
        if (xIndex < mat.length) {
            int path = maxPath(mat, pattern, xIndex + 1, yIndex, 0, sum);
            pathSum = Math.max(pathSum, path);
        }
        if (yIndex < mat[xIndex].length) {
            int path = maxPath(mat, pattern, 0, yIndex + 1, 0, sum);
            pathSum = Math.max(pathSum, path);
        }
    }

    return pathSum;
}
```

שאלה 2 (25 נקודות)

נתון מערך חד-ממדי a **באורך n המלא במספרים שלמים שהם בטווח בין 1 ל- n-1** (אין צורך לבדוק זאת). המערך אינו ממוין!

כתבו שיטה סטטית findDuplicate המקבלת את המערך a כפרמטר ומחזירה מספר שמופיע במערך יותר מפעם אחת.

לדוגמא:

- בהינתן המערך {2, 4, 5, 3, 5, 1} השיטה צריכה להחזיר את הערך 5.
- בהינתן המערך {1, 1, 1, 2, 2, 2, 2} השיטה יכולה להחזיר את הערך 1 או את הערך 2.

חתימת השיטה היא:

```
public static int findDuplicate(int []a)
```

רמז – מותר לשנות את המערך. אין צורך להחזיר אותו למצבו המקורי.

מה סיבוכיות זמן הריצה והמקום של השיטה שכתבתם? הסבירו תשובתכם.

אפשר להניח שהמערך אינו null ואינו ריק, ושכל המספרים בו הם בטווח בין 1 ל-n-1 אין צורך לבדוק זאת.

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.

אל תשכחו לתעד את מה שכתבתם!

```java
public static int findDuplicate(int[] a)
{

    for (int i = 0; i < a.length; i++){
        if (a[0] == a[a[0]]){
            return a[0];
        }
        // swap(0, a[0]);
        int temp = a[0];
        a[0] = a[a[0]];
        a[temp] = temp;
    }
    return 0;
}
}
```

חלק ב - את התשובות לשאלות 3- 5 יש לכתוב על גבי השאלון.
לא נבדוק תשובות שייכתבו במקום אחר!

שאלה 3 (20 נקודות)

נניח שהמחלקה Node שלהלן מממשת עץ 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; }
}
```

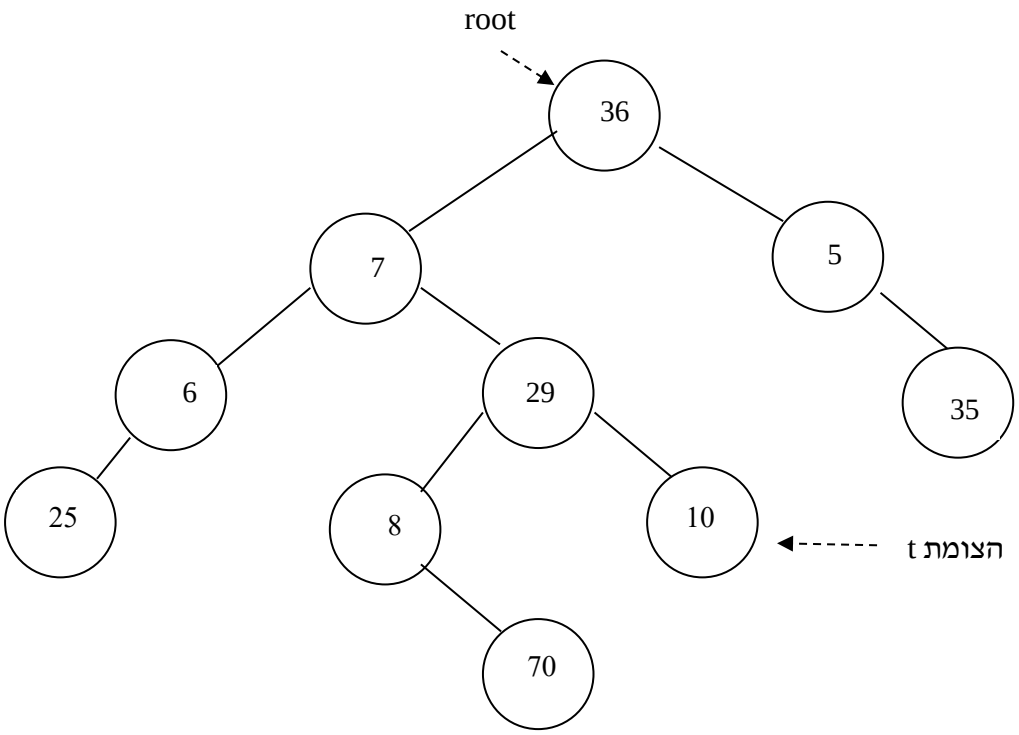
המחלקה BinaryTree מאגדת בתוכה שיטות סטטיות לטיפול **בעץ בינרי**.

בין השיטות נתונות השיטות why ו- what הבאות. שתיהן מקבלות את שורש העץ ועוד צומת אחד שקיים בעץ:

```
    public static Node why(Node root, Node t)
    {
        if (root==null || root.getLeftSon()==t ||
            root.getRightSon()==t)
               return root;
        Node temp = why(root.getLeftSon(),t);
        if (temp==null)
            return why(root.getRightSon(),t);
        return temp;
    }

    public static int what(Node root, Node t)
    {
        if (root == null)
            return 0;
        if(root == why(root,t))
            return 1;
        return 1+ what(root, why(root,t));
    }
```

נתון **העץ הבינרי** הבא, ששורשו הוא root



ענו על הסעיפים הבאים:

(3 נק') (א) מה תחזיר השיטה **why** בעקבות הקריאה BinaryTree.why(root, t)? **דייקו בתשובתכם.**

התשובה היא:

29

(2 נק') (ב) האם יש צומת אחר t1 שהקריאה BinaryTree.why(root, t1) תחזיר אותו ערך שהחזירה בסעיף א? אם כן, כתבו מיהו, אם יש יותר מצומת אחד כזה, כתבו את כולם, אם אין, הסבירו למה לא. **דייקו בתשובתכם.**

התשובה היא:

8

(4 נק') (ג) מה מבצעת השיטה **why** באופן כללי כשהיא מקבלת כפרמטר שורש של **עץ בינרי** root וצומת נוסף t? 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 האם הבן הימני או השמאלי של צומת מסויימת שווה לערך t שמתקבלת בשיטה. במידה וכן , מחזירה את האב. במידה ולא תחזיר NULL

(4 נק') (ד) איזה ערך תחזיר הקריאה BinaryTree.what(root, t)?

התשובה היא:

3

(4 נק') (ה) האם יש צומת אחר t1 שהקריאה BinaryTree.what(root, t1) תחזיר אותו ערך שהחזירה what בסעיף ד? אם כן, כתבו מיהו, אם יש יותר מצומת אחד כזה, כתבו את כולם, אם אין, הסבירו למה לא. **דייקו בתשובתכם.**

התשובה היא:

8

(3 נק') (ו) מה מבצעת השיטה **what** באופן כללי כשהיא מקבלת כפרמטר שורש של **עץ בינרי** root, וצומת נוסף t? 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. **התייחסו למקרי קצה!**

התשובה היא:

השיטה סוכמת את עומק העץ כולל השורש עד למציאת בן שמאלי או ימני ששווה לt .

במידה ואין כזה , השיטה תחזיר את עומק העץ המלא.

במידה ויש כזה , השיטה תחזיר את עומק העץ עד לנקודה t .

שאלה 4 (14 נקודות)

נתונות המחלקות הבאות :

```
public class AAA
{
    private int _val;

    public AAA()  {
        _val = 1;
    }

    public AAA(int val)  {
        _val = val;
    }

    public int getVal()  {
        return _val;
    }

    public void setVal (int val)  {
        _val = val;
    }

    public String toString()  {
        return "_val=" + _val;
    }

} //end of class AAA

-----------------------------------------------------------------
public class BBB extends AAA
{
    private String _st;

    public BBB() {
        _st = "bbb";
    }

    public BBB(String st, int val) {
        super(val);
        _st = st;
    }

    public String getSt() {
        return _st;
    }

    //     המשך המחלקה בעמוד הבא
```

```
    public String toString()
    {
        return "_val= " + getVal()+ " _st= "+_st;
    }

    public boolean equals (BBB other)
    {
        System.out.println("BBB equals");
        return getVal() == other.getVal()  &&
               _st.equals(other.getSt());
    }
} //end of class BBB
```

כמו כן, נתונה באותו פרויקט המחלקה Driver ובתוכה השיטה main הבאה:

```
    public static void main (String[] args)
    {
        AAA a = new AAA(5);
        BBB b = new BBB();
        System.out.println(a);
        System.out.println(b);
    }
```

סעיף א: (2 נקודות)

מה הפלט שיודפס לאחר ביצוע קטע הקוד שבשיטה main?

התשובה היא:

_val = 5

_val = 1 _st = bbb

סעיף ב: (3 נקודות)

1. אם נוסיף בסוף השיטה main את הפקודות הבאות:

a = b;
System.out.println (a.equals(b)); a AAA = new BBB()

מה יודפס כתוצאה מביצוע הפקודה ואיזו גרסה של השיטה equals תופעל?

התשובה היא (הקיפו בעיגול את התשובה הנכונה):

יודפס: false / true

השיטה שתופעל היא: זו שבמחלקה BBB / זו שבמחלקה Object

2. אם נוסיף ל-main לאחר הפקודות שבסעיף ב.1 את הפקודה:

System.out.println (b.equals(a));

מה יודפס כתוצאה מביצוע הפקודה ואיזו גרסה של השיטה equals תופעל?

התשובה היא (הקיפו בעיגול את התשובה הנכונה):

יודפס: true / false

השיטה שתופעל היא: זו שבמחלקה BBB / זו שבמחלקה Object

סעיף ג: (4 נקודות)

הוספנו לשיטה main שבמחלקה Driver את הפקודות הבאות:

```
AAA a2 = new BBB();
BBB b2 = new BBB("bbb", 2);
BBB b3 = new BBB();
```

מה יודפס בעקבות כל אחת מהפקודות הבאות:

1. System.out.println (a2.equals(b3));

התשובה היא:

False

2. System.out.println (b3.equals(a2));

התשובה היא:

False

3. System.out.println (b2.equals(b3));

התשובה היא:

BBB equals false

4. System.out.println (b3.equals(b));

התשובה היא:

BBB equals true

סעיף ד: (5 נקודות)

הוספנו למחלקה AAA שתי שיטות חדשות (ללא קשר לסעיפים הקודמים):

```
public boolean equals (Object obj) {
    _val++;
    AAA a = (AAA) obj;
    return a._val == (_val - 1);
}

public boolean eq(AAA a)  {
    return super.equals(a);
}
```

כמו כן, הוספנו לשיטה main של המחלקה Driver את הפקודות הבאות:

```
AAA a11 = new AAA(2);

AAA a22 = new AAA(2);

Object a33 = a11;
```

מה יקרה/יודפס בעקבות כל אחת מהפקודות הבאות? הקיפו בעיגול את התשובה הנכונה.

1. System.out.println (a11.equals(null));

יודפס true / יודפס false / שגיאת קומפילציה / שגיאת זמן ריצה

2. System.out.println (a11.equals(a22));

יודפס true / יודפס false / שגיאת קומפילציה / שגיאת זמן ריצה

3. System.out.println (a33.equals(a33));

יודפס true / יודפס false / שגיאת קומפילציה / שגיאת זמן ריצה

4. System.out.println (a22.eq(a22)); // שימו לב, כאן נקראה השיטה eq

יודפס true / יודפס false / שגיאת קומפילציה / שגיאת זמן ריצה

5. System.out.println (a11.eq(a33)); // שימו לב, כאן נקראה השיטה eq

יודפס true / יודפס false / שגיאת קומפילציה / שגיאת זמן ריצה

שאלה 5 (16 נקודות)

נתונה המחלקה IntNode הבאה, המייצגת איבר ברשימה מקושרת חד-סטרית המכילה מספרים שלמים:

```
public class IntNode
{
    private int _value;
    private IntNode _next;

    public IntNode(int val, IntNode n) {
        _value = val;
        _next = n;
    }

    public IntNode getNext( )  {
          return _next;
    }
    public int getValue()  {
          return _value;
    }
}
```

נתונה רשימה מקושרת **חד-סטרית**, הממומשת בעזרת המחלקה IntList שלהלן:

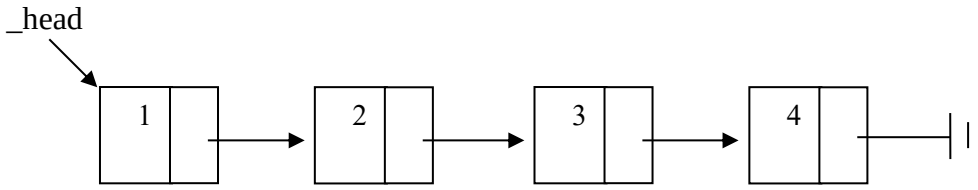
```
public class IntList
{
    private IntNode _head;
    public IntList()     {
        _head = null;
    }

    public int secret (int x)
    {
        int count=0;
        IntNode p = _head, q = _head.getNext();
        while (p!=null && q!=null)
        {
            if (p!=q && q.getValue()- p.getValue()== x)
            {
                System.out.println ("("+p.getValue()+
                                    ","+q.getValue()+")");
                p = p.getNext();
                q = q.getNext();
                count++;
            }
            else if(q.getValue()- p.getValue()< x)
                q = q.getNext();
            else
                p = p.getNext();
        }
        return count;
    }

}    // end of class IntList
```

הניחו שיש במחלקה גם שיטה שמכניסה ערכים לרשימה. אינכם צריכים לדאוג לכך.

בשאלות להלן, נסמן את איברי הרשימה כמספרים מופרדים בפסיקים, בתוך סוגריים מסולסלים. כך לדוגמא, נסמן { 1 , 2 , 3 , 4 } את הרשימה שלהלן (המספר הראשון משמאל הוא המספר שבראש הרשימה):



בהנחה שהרשימה list עליה נפעיל את השיטה secret היא זו:

list = { -6, -3, -2, 0, 1, 3, 7, 8, 10 }

ענו על הסעיפים הבאים:

סעיף א (4 נקודות)

איזה ערך תחזיר הקריאה לשיטה list.secret(4) ? ומה תדפיס השיטה?

התשובה היא:

יוחזר: 3

יודפס (לא בהכרח כל השורות יתמלאו):

-6 , -2

-3 , 1

3 , 7

סעיף ב (4 נקודות)

האם יש x כך שבקריאה list.secret(x) על הרשימה list שלעיל הערך שיוחזר מהשיטה יהיה 2?

אם כן, מה ערכו של x? אם לא, נמקו מדוע.

התשובה היא:

5

סעיף ג (3 נקודות)

מה מבצעת השיטה **secret** באופן כללי כשהיא מופעלת על רשימה חד-סטרית **ממוינת בסדר עולה ממש** (אין מספרים זהים), ומקבלת כפרמטר מספר שלם חיובי x? 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, ומה היא מדפיסה? התייחסו למקרי קצה.**

התשובה היא:

השיטה בודקת זוגות סדורים האם כל זוג שווה למספר איקס שמתקבל בשיטה , במידה וכן הוא ידפיס את הזוג הסדור ויתקדם קדימה במידה והזוג מביא תוצאה קטנה מאיקס אז רק הקיו יתקדם. במידה וגדול מאיקס הצומת תתקדם

סעיף ד (5 נקודות)

1. מהי סיבוכיות הזמן של השיטה secret ?
2. אם לא מובטח שהרשימה ממוינת בסדר עולה ממש, מהי סיבוכיות הזמן הנדרשת לביצוע מה שעושה השיטה secret? נמקו והסבירו את תשובתכם. **אין צורך לכתוב את השיטה שתעשה זאת.**

התשובה היא:

1. O(n(2

2. ((O(N Log(n

ניתן לעשות שימוש בQuick sort

ב ה צ ל ח ה!