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

שאלה 1 (25 נקודות)

כתבו שיטה סטטית רקורסיבית בוליאנית המקבלת מערך של מספרים שלמים arr. השיטה צריכה להחזיר true אם אפשר לחלק את איברי המערך לשתי קבוצות שונות **שוות בגודלן** (כלומר מספר האיברים בהם זהה) כך **שסכום האיברים בשתי הקבוצות שווה**. אם אי אפשר לעשות זאת, השיטה תחזיר false.

חתימת השיטה היא:

```
public static boolean equalSplit (int[] arr)
```

דוגמאות:

- עבור המערך הבא:

0	1	2	3	4	5
-3	5	12	14	-9	13

השיטה תחזיר true כי $-3 + 5 + 14 = 12 + -9 + 13$ וגם בכל קבוצה יש 3 איברים

- עבור המערך הבא:

0	1	2	3	4	5
-3	5	-12	14	-9	13

השיטה תחזיר false כי אמנם $-3 + 5 + 14 + -12 = -9 + 13$ אבל מספר האיברים בשתי הקבוצות שונה (באחת יש ארבעה איברים ובשנייה רק שניים).

- עבור המערך הבא:

0	1	2	3	4
-3	5	-12	14	-9

השיטה תחזיר false כי אין חלוקה של איברי המערך לשתי קבוצות שוות בגודלן.

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

אפשר להשתמש בהעמסת-יתר (overloading). אפשר להניח שהמערך אינו null ואינו ריק.

אין לשנות את תוכן המערך (אפילו לא זמנית), ולא להשתמש במערך עזר.

אין צורך לדאוג ליעילות השיטה! אבל כמובן שצריך לשים לב לא לעשות קריאות רקורסיביות מיותרות!

אל תשכחו לתעד את מה שכתבתם!

```java
public static boolean equalSplit(int[] arr)
{
    return equalSplit(arr,0,0,0,0,arr.length,0);
}

private static boolean equalSplit(int[] arr,int sum1, int sum2, int cnt1, int cnt2,int left, int i) {
    if (arr.length % 2 != 0) // if arr length is not double ( 3/3 2/2 ZUGI )
        return false;
    if (left == 0 && cnt1 == cnt2 && sum1 == sum2)
        return true;
    if (left == 0 && (cnt1 != cnt2 || sum1 != sum2))
        return false;
    return equalSplit(arr,sum1+arr[i],sum2,cnt1+1,cnt2,left-1,i+1)
        || equalSplit(arr,sum1,sum2+arr[i],cnt1,cnt2+1,left-1,i+1);
}
```

שאלה 2 (25 נקודות)

בשאלה זו נתייחס למערכים דו-ממדיים ריבועיים, כלומר, מספר השורות והעמודות שווה (נניח שהוא שווה ל-n).

לצורך השאלה נניח כי n הוא חזקה שלמה של 2.

עבור מערך כזה, נגדיר חלוקה פנימית שלו לארבעה רובעים בגודל n/2 × n/2, ממוספרים מ-1 עד 4 באופן הבא:

1	2
4	3

נאמר שהמערך הוא **סיבובי** (circular) אם כל האיברים ברובע 1 קטנים ממש מכל אלו שברובע 2, אלו שברובע 2 קטנים ממש מכל אלו שברובע 3, ואלו שברובע 3 קטנים ממש מכל אלו שברובע 4.

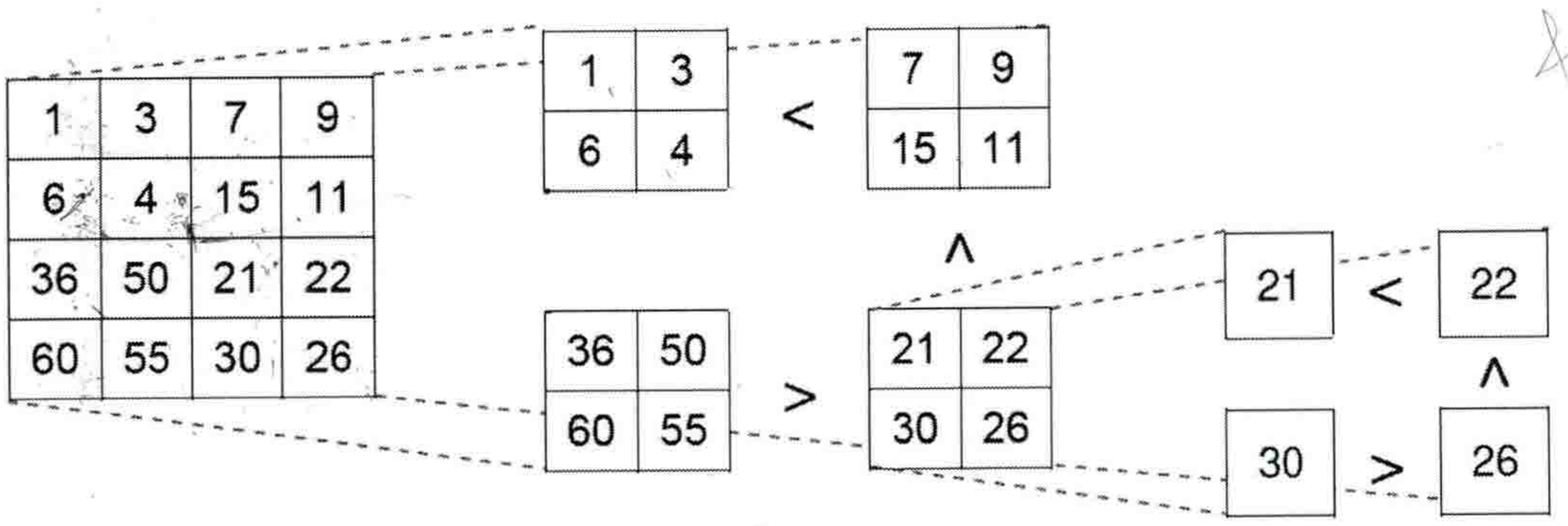
למשל, המערך הבא הוא סיבובי:

1	5
9	7

לשם הנוחות, נגדיר גם כל מערך בגודל 1×1 כמערך סיבובי.

נאמר שמערך בגודל n×n הוא **ממוין-סיבובית** (circular sorted) אם הוא סיבובי, ארבעת הרובעים שלו סיבוביים, וכן הלאה עד לרובעים בגודל 1×1.

לדוגמא, המערך הבא ממוין-סיבובית:



עליכם לכתוב שיטה סטטית בוליאנית לחיפוש במערך ממוין-סיבובית. השיטה מקבלת כפרמטרים את מערך דו-ממדי mat שהוא ממוין-סיבובית, וערך לחיפוש num. אם הערך num נמצא במערך mat, השיטה תחזיר true ותדפיס את מספר השורה ומספר העמודה שבהם נמצא המספר num. אם הערך num לא נמצא במערך mat, השיטה תחזיר false ולא יודפס כלום.

חתימת השיטה היא:

```
public static boolean search (int [][] mat, int num)
```

```java
public static boolean search (int [][]mat,int num)
{
    int len = mat.length;
    int row = len-1;
    int col = 0;
    int value = num-1;
    int firstQuarter,secondQuarter,thirdQuarter,fourthQuarter;
    int halfSize = len/2;
    if(row == 0) {
        if(num == mat[row][col]) {
        System.out.println("row=0\ncol=0");
        return true;
        }
    return false;
    }

    while(num != value && halfSize > 0){
        firstQuarter=mat[row-halfSize][col];
        secondQuarter=mat[row-halfSize][col+halfSize];
        thirdQuarter=mat[row][col+halfSize];
        fourthQuarter=mat[row][col];

        if(firstQuarter >= num) {
            value=firstQuarter;
            row-=halfSize;
        }
        else if(secondQuarter >= num) {
            value=secondQuarter;
            row-=halfSize;
            col+=halfSize;
        }
        else if(thirdQuarter >= num) {
            value=thirdQuarter;
            col+=halfSize;
        }
        else if(fourthQuarter >= num) {
            value=fourthQuarter;
        }else{
            return false;
        }
        halfSize/=2;
    } // while
    if(value == num) {
        System.out.println("row=" + row+ "\ncol= " +col);
        return true;
    }
    return false;

}
```

לדוגמא,

אם המערך mat הוא המערך המצויר לעיל, והמספר num הוא 22, השיטה תחזיר את הערך true, ויודפסו השורות הבאות:

row = 2

col = 3

אם המערך mat הוא המערך המצויר לעיל, והמספר num הוא 23, השיטה תחזיר את הערך false, ולא יודפס כלום.

אתם יכולים להניח שהמערך mat אינו null והוא ממוין-סיבובית. אינכם צריכים 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.

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

אל תשכחו לתעד את מה שכתבתם!

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

שאלה 3 (18 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**.

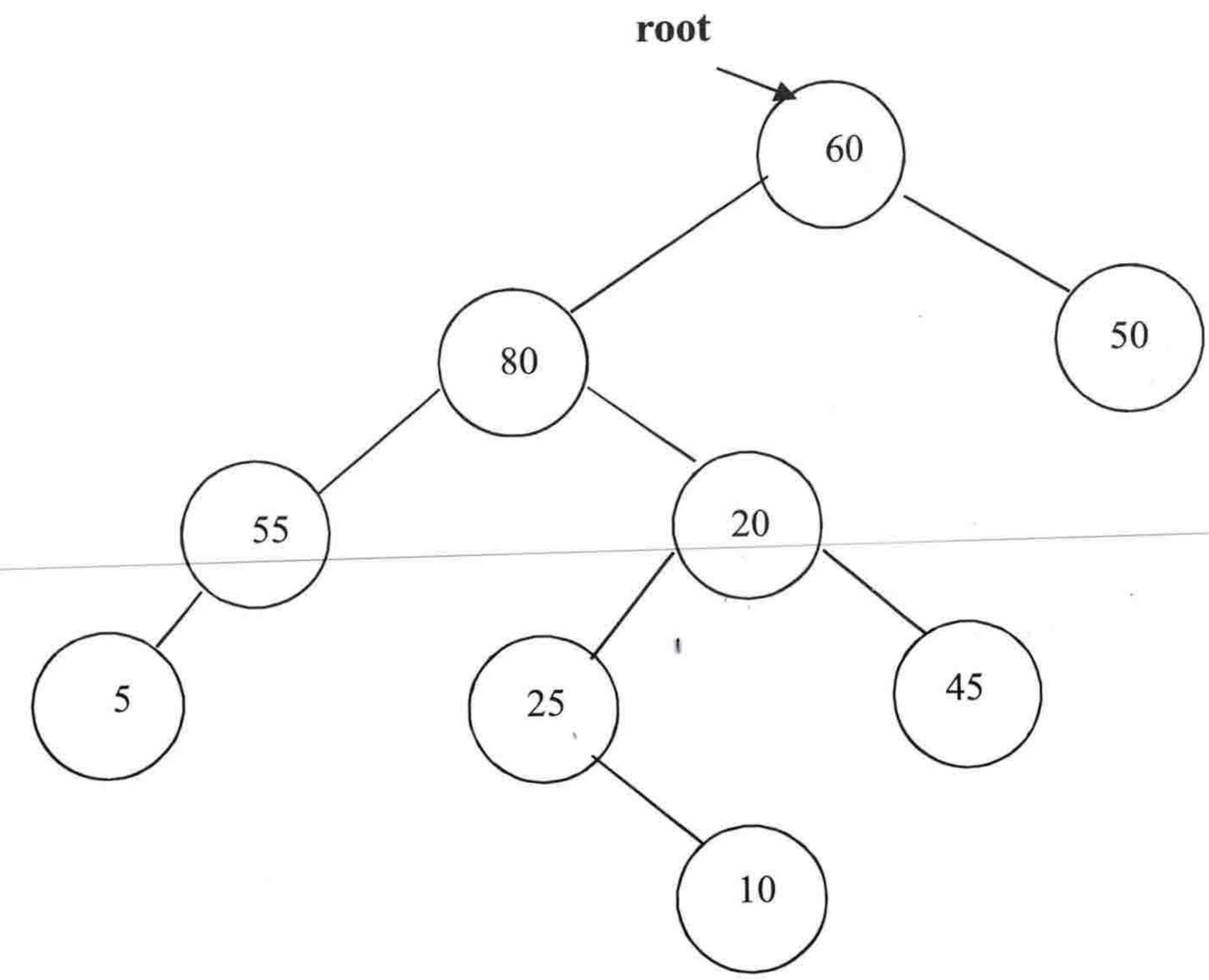
בין השיטות נתונה השיטה printSecret הבאה, המקבלת שורש של עץ בינרי ומספר שלם כלשהו.

```
public static boolean printSecret(Node node, int target)
{
     if (node == null)
         return false;
     if (node.getNumber() == target)
         return true;

     if (printSecret(node.getLeftSon(), target)
     || printSecret(node.getRightSon(), target))
     {
         System.out.print(node.getNumber() + " ");
         return true;
     }

     return false;
}
```

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



ענו על הסעיפים הבאים:

(3 נק') א. בעקבות הקריאה BinaryTree.printSecret(root, 40), מה יודפס ואיזה ערך תחזיר השיטה **printSecret**?

התשובה היא:

הערך שיוחזר הוא: False

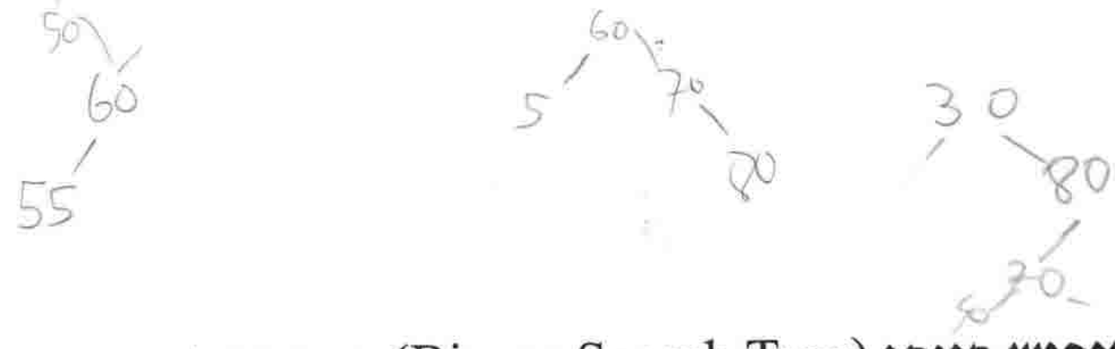
יודפס (משמאל לימין): כלום

(4 נק') ב. בעקבות הקריאה BinaryTree.printSecret(root, 10), מה יודפס ואיזה ערך תחזיר השיטה **printSecret**?

התשובה היא:

הערך שיוחזר הוא: True

יודפס (משמאל לימין): 25 20 80 60

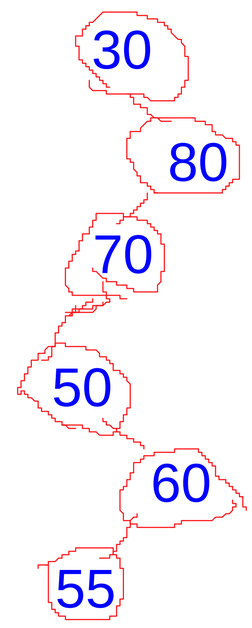


(5 נק') ג. עליכם לצייר **עץ חיפוש בינרי** (Binary Search Tree) ששורשו root כך שאם נקרא לשיטה printSecret עם השורש root והמספר 55 יוחזר הערך true ויודפס הפלט (משמאל לימין):

60 50 70 80 30

אם לא קיים עץ חיפוש בינרי כזה, עליכם להסביר מדוע.

התשובה היא:



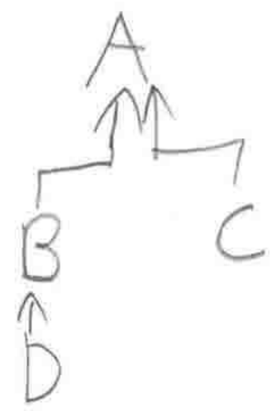
(6 נק') ד. מה מבצעת השיטה **printSecret** באופן כללי כשהיא מקבלת כפרמטר שורש של **עץ בינרי** root ומספר כלשהו target? שימו לב, עליכם לתת תיאור ממצה של **מה עושה השיטה באופן כללי**, ולא תיאור של מה עושה כל שורה בשיטה, או איך היא מבצעת זאת. **כלומר, מה המשמעות של הערך שהשיטה מחזירה, ומה המשמעות של ההדפסה? התייחסו למקרי קצה!**

התשובה היא:

השיטה בודקת האם עלה שווה לטרגט , במידה וכן מחזיר אמת
ולאחר מכן מחזירה את כל העלים בדרך לעלה שנמצא בעץ
.זאת אומרת מהשורש עד לעלה טרגט לא כולל

שאלה 4 (16 נקודות)

נתון פרויקט שהוגדרו בו המחלקות שלהלן. כל אחת בקובץ נפרד, כמובן.




```
public class A {
    protected int _x;

    public boolean something(Object obj) {
        System.out.println("1");
        return super.equals(obj);
    }
}
//-------------------------------------------------------------//
public class B extends A {

    public boolean something(Object obj) {
        System.out.println("2");
        return super.something(obj);
    }

    public boolean something(B obj) {
        System.out.println("3");
        return super.something(obj);
    }

    public boolean something(A obj) {
        System.out.println("4");
        return super.something(obj);
    }
}
//-------------------------------------------------------------//
public class C extends A {

    public boolean something(B obj) {
        System.out.println("5");
        return super.something(obj);
    }
}
//-------------------------------------------------------------//
```

```
public class D extends B {

      public boolean something(A obj) {
            System.out.println("6");
            return super.something(obj);
      }

      public boolean something(B obj) {
            System.out.println("7");
            return super.something(obj);
      }

      public boolean something(D obj) {
            System.out.println("8");
            return super.something(obj);
      }
}
```

```
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();
            D d2 = new D();
            A d3 = new D();


                  // כאן יופיעו הפקודות //

      }
}
```

להלן נתונות 8 פקודות. התייחסו אליהן כאילו הן נמצאות בשיטה main וכתבו מה קורה לאחר כל פקודה.

אם הפקודה לא עוברת קומפילציה, כתבו למה. אם יש שגיאה בזמן ריצה, כתבו למה.

אם הכל תקין, כתבו מה יהיה הפלט שיודפס בעקבות הפקודה.

אין קשר בין הפקודות!

1. `b1.something(a2);`

   2
   1

2. `c1.something(d2);`

   5
   1

3. `d1.something(b1);`

   7
   3
   1

4. `a2.something(c1);`

   Compilation Error

5. `b2.something(d3);`

   2
   1

6. `c2.something(c2);`

   1

7. `d2.something(a1);`

   6
   4
   1

8. `d3.something(c2);`

   2
   1

(4)

שאלה 5 (16 נקודות)

נתונה המחלקה IntNode הבאה, המייצגת איבר ברשימה:

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

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

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

      public void setValue(int v)  {
            _value = v;
      }
      public void setNext(IntNode node)  {
            _next = node;
      }
}
```

נתונה רשימה מקושרת של מספרים שלמים, הממומשת בעזרת המחלקה IntList שלהלן:

```
public class IntList
{
    private IntNode _head;
    public IntList( ) {
        _head = null;
    }
    public IntList(IntNode h ) {
        _head = h;
    }
            //  כאן יש עוד בנאים ושיטות...  //

            //    המשך המחלקה בעמוד הבא    //
```

```
public int f()
{
    IntNode temp = _head;
    int c = 0;
    while (temp != null)
    {
        c++;
        temp = temp.getNext();
    }
    return c;
}

public int secret2021B62(int x)
{
    int curr = 0, result = f()+1;
    int temp1 = 0, temp2 = 0;
    IntNode ptr1 = _head, ptr2 = _head;
    while (ptr2!= null)
    {
        while (curr <= x && ptr2!= null)
        {
            curr += ptr2.getValue();
            ptr2 = ptr2.getNext();
            temp2++;
        }

        while (curr > x && ptr1 !=null)
        {

            if (temp2 - temp1 < result)
                result = temp2 - temp1;

            curr -= ptr1.getValue();
            ptr1 = ptr1.getNext();
            temp1++;
        }
    }
    return result;
}

. . .               // other methods

}
```

אתם יכולים להניח שהרשימה מלאה במספרים שלמים אי שליליים בלבד!

בטענות להלן, נסמן את איברי הרשימה כמספרים מופרדים בפסיקים, בתוך סוגריים מסולסלים.

כך לדוגמא, נסמן { 21 , 12 , 9 , -4 } את הרשימה null ← 21 ← 12 ← 9 ← -4

סעיף א (1 נקודות)

מה הערך אותו תחזיר השיטה **f** כשנפעיל אותה על הרשימה {3, 6, 2, 7}?

התשובה היא:

4

סעיף ב (2 נקודות)

מה מבצעת השיטה **f** באופן כללי? הסבירו בקצרה **מה** השיטה עושה ולא **כיצד** היא מבצעת זאת.
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, עליכם לכתוב מה המשמעות של הערך שמוחזר מהשיטה f, כשהיא מופעלת על רשימה כלשהי.

התשובה היא:

סוכמת את איברי הרשימה

לגבי סעיפים ג – ה להלן נתונה הרשימה { 19 , 0 , 6 , 13 , 4 , 1 } = list.

סעיף ג (2 נקודות)

איזה ערך תחזיר הקריאה לשיטה `list.secret2021B62(22)?`

התשובה היא:

3

סעיף ד (2 נקודות)

איזה ערך תחזיר הקריאה לשיטה `list.secret2021B62(43)?`

התשובה היא:

7

(המשך השאלה בעמוד הבא)

סעיף ה (2 נקודות)

איזה ערך תחזיר הקריאה לשיטה `?list.secret2021B62(2)`

התשובה היא:

1

סעיף ו (7 נקודות)

מה מבצעת השיטה **secret2021B62** באופן כללי כשהיא מופעלת על רשימה מקושרת **שכל איבריה אי-שליליים**, ומקבלת כפרמטר מספר אי-שלילי x? הסבירו בקצרה **מה** השיטה עושה ולא **כיצד** היא מבצעת זאת.

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 שיוחזר כתוצאה מהפעלת השיטה על רשימה מקושרת כלשהי **שכל איבריה אי-שליליים** ומספר x אי-שלילי.

התייחסו למקרי קצה.

התשובה היא:

השיטה מעבירה את אורך התת רשימה הקצרה ביותר שסכום הערכים של איבריה גדול ממש מ- x.
אם סכום כל הערכים של איברי הרשימה (כל הרשימה) לא גדול מ- x, יוחזר אורך הרשימה הכולל + 1.
אם הרשימה ריקה, יוחזר 1 (באמת).

5

בהצלחה!!