

פיתוח:

משתתפי קורס מורים מובילים תשע"ה

התאמת שאלות לתכנית הלימודים החדשה
מבני נתונים

תוכן עניינים

בגרות תשנ"ז שאלה 1 (ברכה דאום-רייטר)

בגרות תשנ"ח שאלה 13 (דפנה מינסטר, ולרי פקר, מושית ולץ ורחל בן עמי)

בגרות תשס"א שאלה 1 (שרה פונק)

בגרות תשס"ג שאלה 1 (חיה עידן)

בגרות תשס"ה שאלה 2 (יבגני קנל ודורית ליקרמן)

בגרות תשס"ו שאלה 4 (ענת שלוש סגל, אביטל גרינולד, מושית ולץ ורחל בן עמי)

בגרות תשס"ז שאלה 4 (רוני אלנקרי, יוסי זהבי)

בגרות תשס"ח שאלה 2 (רחל לודמור, גיא אורן)

בגרות תשס"ט שאלה 1 (ולרי פקר, דפנה לוי רשתי)

בגרות תש"ע שאלה 2 (גלית שריקי, חני טוראל, ישראל אברמוביץ)

בגרות תשע"ג שאלה 1 (ולרי פקר, רחלי צרניחוב, משה ניסים)

בגרות תשע"ג שאלה 3 (ולרי פקר)

ניסוח מחודש של שאלות בגרות והתאמתן לתכנית הלימודים החדשה

פיתוח : משתתפי קורס מורים מובילים תשע"ה

שאלה 1, בגרות תשנ"ז 1997

ברכה דאום-רייטר

הניסוח המקורי:

נתונות שתי רשימות מקושרות L1 ו-L2, של מספרים שלמים גדולים מאפס שהאורכים שלהם לא ידועים. כתוב תכנית, בשפת פסקל או בשפת בייסיק מובנה, שתיצור משתי הרשימות L1 ו-L2 רשימה מקושרת חדשה L3 באופן הזה: התכנית תאתר עבור כל איבר ב-L1 את האיברים ב-L2 שהם כפולות שלו, ותעביר את האיברים האלו ל-L3. כלומר, בסוף התהליך תכיל הרשימה L2 אך ורק איברים שלא נמצאו להם מחלקים ב-L1.

הצעה לניסוח חדש שייתאים לתכנית "מבני נתונים":

נתונות הפניות לשתי שרשרות חוליות chain1 ו-chain2, שבהן מספרים שלמים גדולים מאפס שהאורכים שלהן לא ידועים.

כתוב פעולה חיצונית שתקבל את שתי ההפניות לשרשרות chain1 ו-chain2 ותחזיר הפניה לשרשרת חוליות חדשה chain3 שתיבנה באופן הזה:

הפעולה תאתר עבור כל איבר ב-chain1 את האיברים ב-chain2 שהם כפולות שלו, ותעביר את האיברים הללו ל-chain3. כלומר בסוף התהליך תכיל השרשרת chain2 אך ורק איברים שלא נמצאו להם מחלקים ב-chain1.

הצעה נוספת:

1. נתונה כותרת הפעולה:

```
(public static Boolean IsDivider(Node<int> N1, Node<int> N2)
```

המקבלת הפניות לשתי חוליות ומחזירה אמת אם החוליה N1 מחלקת את החוליה N2.

ממש את הפעולה בשפת java.

2. נתונות הפניות לשתי שרשרות חוליות chain1 ו-chain2, של מספרים שלמים גדולים מאפס שהאורכים שלהם לא ידועים.

כתוב פעולה חיצונית שתקבל את שתי ההפניות לשרשרות chain1 ו-chain2 ותחזיר הפניה לשרשרת חוליות חדשה chain3 שתיבנה באופן הזה:

הפעולה תאתר עבור כל איבר ב-chain1 את האיברים ב-chain2 שהם כפולות שלו, ותעביר את האיברים הללו ל-chain3. כלומר בסוף התהליך תכיל השרשרת chain2 אך ורק איברים שלא נמצאו להם מחלקים ב-chain1.

השתמש בפעולת העזר שכתבת בסעיף 1.

שאלה 13, בגרות קיץ תשנ"ח 1998

הניסוח המקורי:

מקטע-משותף-מקסימלי לשתי רשימות נתונות הוא תת-הרשימה הרצופה המקסימלית המשותפת לשתייהן.

לדוגמה: עבור הרשימות L1 ו-L2 :

L1 = 1, 5, 6, 3, 4, 8, 9, 5, 4, 3, 6, 7

L2 = 5, 4, 4, 8, 9, 5, 6, 3, 6

המקטע-המשותף-המקסימלי הוא הרשימה L3 : L3 = 4, 8, 9, 5

נתונה הפונקציה:

function list_compare (L1, L2 : list_type; pos1, pos2 : pos_type; n : integer) : boolean;

הפונקציה מקבלת שתי רשימות L1 ו-L2 ושני מצביעים pos1 ו-pos2 המצביעים על מקומות ב-L1 ו-L2 בהתאמה. הפונקציה מחזירה TRUE אם n האיברים החל מ-pos1 ברשימה L1 זהים ל-n האיברים החל מ-pos2 ברשימה L2, ו-FALSE אחרת.

אם מספר האיברים אחרי pos1 או pos2 קטן מ-n, תחזיר הפונקציה FALSE.

א. כתוב פרוצדורה בסביבת העבודה, שתקבל שתי רשימות L1 ו-L2

ותחזיר את המקטע-המשותף-המקסימלי לשתייהן.

ב. מהי סיבוכיות זמן הריצה של הפעולה שמימשת בסעיף הקודם? נמק.

אם נעזרת בפונקציה list_compare, הנח כי סיבוכיות זמן הריצה שלה,

כפונקציה של n המועבר כפרמטר, היא: O(n).

הצעה 1 של רחל בן-עמי ומושית ולץ

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

מקטע_משותף_מקסימלי לשתי רשימות נתונות הוא תת הרשימה הרצופה המקסימלית, המשותפת לשתייהן.

לדוגמה, עבור הרשימות L1 ו-L2, כאשר L1, L2 מטיפוס Node<int>.

L1 → 1, 5, 6, 3, 4, 8, 9, 5, 4, 3, 6, 7

L2 → 5, 4, 4, 8, 9, 5, 6, 3, 6

המקטע_המשותף_המקסימלי הוא הרשימה : L3 → 4, 8, 9, 5

נתונה הפעולה:

public static bool ListCompare(Node<int> L1, Node<int> L2, Node<int> pos1,

Node<int> pos2, int n)

הפעולה מקבלת שתי רשימות L1, L2 ושתי הפניות pos1, pos2 למקומות ב-L1, L2 בהתאמה.

הפעולה מחזירה TRUE אם n האיברים החל מ-pos1 ברשימה L1 זהים ל-n האיברים החל מ-pos2

ברשימה L2, ו-FALSE אחרת.

אם מספר האיברים אחרי pos1 או pos2 קטן מ- n , תחזיר הפעולה FALSE.

א. כתוב פעולה שתקבל שתי רשימות L1 ו-L2 מטיפוס `Node<int>`

ותחזיר את המקטע **המשותף_המקסימלי** לשתייהן.

ב. מהי סיבוכיות זמן הריצה של הפעולה שמימשת בסעיף הקודם? נמק.

אם נעזרת בפעולה `ListCompare`, הנח כי סיבוכיות זמן הריצה שלה, כפונקציה של n המועבר

כפרמטר, היא $O(n)$.

הצעה 2 של דפנה מינסטר

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

מקטע-משותף-מקסימלי לשני אוספים נתונים הוא תת-אוסף הרצוף המקסימלי המשותף לשניהם.

לדוגמה: עבור האוספים collect1 ו-collect2

collect1 ⇒ 1 ⇒ 5 ⇒ 6 ⇒ 3 ⇒ 4 ⇒ 8 ⇒ 9 ⇒ 5 ⇒ 4 ⇒ 3 ⇒ 6 ⇒ 7 ⇒ ||

collect2 ⇒ 5 ⇒ 4 ⇒ 4 ⇒ 8 ⇒ 9 ⇒ 5 ⇒ 6 ⇒ 3 ⇒ 6 ⇒ ||

המקטע-המשותף-המקסימלי הוא האוסף collect3:

collect3 ⇒ 4 ⇒ 8 ⇒ 9 ⇒ 5 ⇒ ||

נתונה השיטה:

```
public static boolean collectCompare ( Node < Integer > collect1 ,  
                                       Node < Integer > collect2 ,  
                                       Node < Integer > p1 ,  
                                       Node < Integer > p2 , int N )
```

השיטה מקבלת שני אוספים מאותחלים collect1 ו-collect2 ושני מצביעים (מיקומים) p1 ו-p2 המצביעים על

מקומות ב-collect1 ו-collect2 בהתאמה. השיטה מחזירה 'אמת' (true) אם האיברים החל מ-p1 באוסף

collect1 זהים ל-N האיברים החל מ-p2 באוסף collect2, אחרת מחזירה 'שקר' (false).

אם מספר האיברים אחרי p1 או אחרי p2 קטן מ-N תחזיר השיטה 'שקר' (false).

א. כתבי שיטה בשפת Java שתקבל שני אוספים collect1 ו-collect2 ותחזיר את **המקטע-המשותף-**

המקסימלי לשניהם. השתמש/י בשיטה `collectCompare` (הערה: אין צורך לממש את השיטה).

ב. מהו סדר הגודל של השיטה שכתבת בסעיף א' ? נמקי !

אם נעזרת בשיטה `collectCompare` הנחי כי סדר הגודל שלה, כפונקציה של n המועבר כפרמטר,

הוא: $O(n)$.

הצעה 3 של ולרי פקר (כולל התייחסות להצעה של דפנה מינסטר)

- לדעתי, הניסוח מחדש לשאלה (שהציעה דפנה מינסטר) לא מתאים לתכנית הלימודים החדשה.
1. החוליה היא אבן יסוד למבנה נתונים דינמי ולא אוסף. אוסף הוא מחלקה סגורה התומכת בפעולות הכנסה, הוצאה, סריקה, חיפוש וכדומה.
 2. בתכנית הישנה המבוססת על שפת Pascal היו שני מימושים של טיפוס הנתונים רשימה: באמצעות מערך ובעזרת שרשרת חוליות. בשני ייצוגים של הרשימה בממשק של יחידות הספרייה לא היו הבדלים בהגדרת הפעולות, והן היו זהות לחלוטין. משתמש חיצוני, שכתב קוד הנעזר ביחידת הספרייה אחת, היה יכול להחליף את יחידת הספרייה מבלי לשנות דבר בקוד.
- הפעולות כמו סוף-רשימה(L), עוקב-ברשימה(L, p), עוגן-רשימה(L) היו צריכים רשימה כפרמטר. לכן בניסוח השאלה בשנת 1998 השתמשו בשני הפרמטרים L1 ו-L2. בתוכנית החדשה הייצוג של שרשרת חוליות "פתוח" וידוע מראש ואין צורך בפרמטרים כאלו.

1. החוליה היא אבן יסוד למבנה נתונים דינמי ולא אוסף.
אוסף הוא מחלקה סגורה התומכת בפעולות הכנסה, הוצאה, סריקה, חיפוש וכדומה.

2. בתכנית הישנה המבוססת על שפת Pascal היו שני מימושים של טיפוס הנתונים רשימה:
באמצעות מערך ובעזרת שרשרת חוליות.
בשני ייצוגים של הרשימה בממשק של יחידות הספרייה לא היו הבדלים בהגדרת הפעולות, והן היו זהות לחלוטין. משתמש חיצוני, שכתב קוד הנעזר ביחידת הספרייה אחת, היה יכול להחליף את יחידת הספרייה מבלי לשנות דבר בקוד.

הפעולות כמו סוף-רשימה(L), עוקב-ברשימה(L, p), עוגן-רשימה(L) היו צריכים רשימה כפרמטר.
לכן בניסוח השאלה בשנת 1998 השתמשו בשני הפרמטרים L1 ו-L2.
בתוכנית החדשה הייצוג של שרשרת חוליות "פתוח" וידוע מראש ואין צורך בפרמטרים כאלו.

לכן ניסוח השאלה לדעתי צריך להיות:

מקטע-משותף-מקסימלי לשני שרשרות חוליות נתונות הוא תת-שרשרת החוליות הרצופה המקסימלית, המשותפת לשתייהן. לדוגמה: עבור שרשרות החוליות $chain1$ ו- $chain2$

$chain1 \Rightarrow 1 \Rightarrow 5 \Rightarrow 6 \Rightarrow 3 \Rightarrow 4 \Rightarrow 8 \Rightarrow 9 \Rightarrow 5 \Rightarrow 4 \Rightarrow 3 \Rightarrow 6 \Rightarrow 7 \Rightarrow ||$

$chain1 \Rightarrow 5 \Rightarrow 5 \Rightarrow 5 \Rightarrow 8 \Rightarrow 9 \Rightarrow 5 \Rightarrow 6 \Rightarrow 3 \Rightarrow 6 \Rightarrow ||$

המקטע-המשותף-המקסימלי הוא שרשרת החוליות $chain3$:

$chain3 \Rightarrow 4 \Rightarrow 8 \Rightarrow 9 \Rightarrow 5 \Rightarrow ||$

מציאת מקטע-משותף-מקסימלי, המשותפת לשתי שרשרות החוליות. כדי לאתר את תת-שרשרת החוליות הרצופה המקסימלית החל מחוליות מסוימות בשתי שרשרת החוליות, לא נצטרך להעביר את השרשרות "המלאות" כפרמטרים לפעולה, כיוון שממילא אין צורך לחפש בשרשרות "המלאות". די יהיה בהעברת הפניות לחוליות שמהן אנו רוצים להתחיל את החיפוש. לכן ניסוח השאלה לדעתי צריך להיות:

לפניכם חתימת הפעולה החיצונית:

```
public static boolean collectCompare (Node < Integer > chain1 ,
                                     Node < Integer > chain2 , int n)
```

הפעולה מקבלת שתי הפניות על שרשרות החוליות *chain1* ו-*chain2* ומספר *n*. הפעולה מחזירה 'אמת' אם *n* האיברים החל מהחוליה *chain1* זהים ל-*n* האיברים החל מהחוליה *chain2*, אחרת מחזירה 'שקר'. אם מספר האיברים החל מהחוליה *chain1* או החל מהחוליה *chain2* קטן מ-*n* תחזיר הפעולה 'שקר'.

א. כתבו פעולה בשפת *Java* שתקבל שתי הפניות על שרשרות החוליות *chain1* ו-*chain2* ותחזיר את המקטע-המשותף-המקסימלי לשניהם.

השתמשו בפעולה *collectCompare* בלי לממש אותה

ב. מהו סדר הגודל של הפעולה שכתבת הסעיף א' ? נמק!

אם נעזרת בפעולה *collectCompare* הנחו כי סדר הגודל שלה, כפונקציה של *n* המועבר כפרמטר, הוא $O(n)$.

פתרון אפשרי בשפת C#:

```
public static bool Compare2Nodes(Node<int> chain1, Node<int> chain2, int n)
{
    while (chain1 != null && chain2 != null & n > 0)
    {
        if (chain1.GetValue() != chain2.GetValue())
            return false;
        chain1 = chain1.GetNext();
        chain2 = chain2.GetNext();
        n--;
    }
    return n == 0;
}
```

```

public static Node<int> MaximumSharedSegment(Node<int> chain1, Node<int> chain2)
{
    Node<int> p1 = chain1;
    Node<int> p2, pMax = null;
    int maxSement = 0, n;
    while (p1 != null)
    {
        p2 = chain2;
        while (p2 != null)
        {
            n = maxSement;
            while (Compare2Nodes(p1, p2, n + 1))
                n++;
            if (n > maxSement)
            {
                maxSement = n;
                pMax = p1;
            }
            p2 = p2.GetNext();
        }
        p1 = p1.GetNext();
    }
    if (pMax != null)
    {
        p2 = new Node<int>(pMax.GetValue());
        pMax = pMax.GetNext();
        p1 = p2;
        for (int i = 2; i <= maxSement; i++)
        {
            p1.SetNext(new Node<int>(pMax.GetValue()));
            p1 = p1.GetNext();
        }
    }
}

```

```

        pMax = pMax.getNext();
    }
}
else
    p2 = null;
return p2;
}

```

שאלה 1, בגרות תשס"א 2001

שרה פונק

הניסוח המקורי

לפניך האלגוריתם סוד (הנעזר באלגוריתמים תעלומה והחלף)

סוד (L, x)

{האלגוריתם מקבל רשימת מספרים שלמים שונים L ומספר שלם x , ומחזיר מספר שלם}

(1) עוקב_ברשימה (עוגן_רשימה (L, L)) $P1 \leftarrow$

(2) קודם_ברשימה (סוף_רשימה (L, L)) $P2 \leftarrow$

(3) החזר תעלומה $(L, P1, P2, x)$

תעלומה $(L, P1, P2, x)$

(1) אם $P1 = P2$ אזי

(1.1) אם אחזר_מרשימה $(L, P1)$ אזי

(1.1.1) החזר 1

(1.2) אחרת

(1.2.1) החזר 0

(2) אחרת

(2.1) אם אחזר_מרשימה $(L, P1)$ אזי

(2.1.1) החזר תעלומה $(P2, x)$, עוקב_ברשימה $(L, P1)$ + 1

(2.2) אחרת

(2.2.1) אם אחזר_מרשימה $(L, P2)$ אזי

(2.2.1.1) החזר תעלומה (x) , קודם_ברשימה $(L, P2)$, $(L, P1)$

(2.2.2) אחרת

(2.2.2.1) החלף $(L, P1, P2)$

(2.2.2.2) החזר תעלומה $(L, P1, P2, x)$

החלף (L, P1, P2)

(1) אחזר_מרשימה (L, P1) ← TEMP

(2) עדכן_רשימה (אחזר_מרשימה (L, P1, (L, P2)

(3) עדכן_רשימה (L, P2, TEMP)

א. מה יחזיר האלגוריתם סוד (L, 9), אם יקבל את הרשימות ii – i שלפניך?
פרט את המעקב אחר ביצוע האלגוריתם עבור כל אחת מהרשימות.

i. 8, 25, 9, 3, 41, 10

ii. 7, 2, 9, 5, 24, 7, 18, 3, 14, 4

ב. הסבר באופן כללי מה מבצע האלגוריתם סוד (L, x)

ג. מהי סיבוכיות זמן הריצה של האלגוריתם סוד (L, x) ? נמק.

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

המרת שאלה אלגוריתמית לשאלת מעקב הכתובה ב- java

```
import java.util.Scanner;
public class TestBagrut2001_q1
{
    /**
     * @param args
     */
    public static Node<Integer> inputInt(int num)
    {
        Scanner sc = new Scanner (System.in);
        System.out.println("Enter "+num+" integers");
        Node<Integer> chain = new Node<Integer>(sc.nextInt());
        Node<Integer>pos = chain ;
        for(num-- ; num>0 ;num--, pos = pos.getNext())
            pos.setNext(new Node<Integer>(sc.nextInt()));
        return chain ;
    }
}
```

```

public static <T> void printList(Node<T> head)
{
    while (head!=null)
    {
        System.out.print(head.getValue()+" --> ");
        head = head.getNext();
    }
    System.out.println("null");
}

public static void main(String[] args)
{
    Node<Integer> list1 = inputInt(6);
    printList(list1);
    System.out.println(Bagrut2001_q1.mys1(list1, 0));
}
}

public class Bagrut2001_q1
{
    public static int mys1(Node<Integer> head , int x )
    {
        Node<Integer> p1 = head ;
        Node<Integer> p2 = head ;
        while (p2.getNext()!=null)
            p2 = p2.getNext();
        return mys2(head , p1,p2, x);
    }

    public static int mys2(Node <Integer> head , Node<Integer> p1,
        Node<Integer> p2, int x)

```

```

{
    if(p1 == p2)
        if(p1.getValue() > x)
            return 1 ;
        else
            return 0 ;
    else
        if(p1.getValue() > x)
            return mys2(head, p1.getNext(), p2, x) + 1 ;
        else
            if(p2.getValue() <= x)
            {
                Node<Integer> prev = p1 ;
                while(prev.getNext() != p2)
                    prev = prev.getNext();
                return mys2(head, p1, prev, x);
            }
            else
            {
                int temp = p1.getValue();
                p1.setValue(p2.getValue());
                p2.setValue(temp);
                TestBagrut2001_q1.printList(head);
                return mys2(head, p1, p2, x);
            }
    }
}

```

```

public class Node <T>
{
    private T value ;
    private Node<T> next ;
    public Node (T x )

```

```
{
    this.value = x ;
    this.next = null;
}

public Node (T x , Node<T> next )
{
    this.value = x ;
    this.next = next;
}

public T getValue(){return this.value ;}
public Node<T> getNext () {return this.next ;}
public void setValue(T x){ this.value = x ;}
public void setNext (Node<T> next){ this.next = next ;}
public boolean hasNext(){return this.next == null;}
public String toString(){return this.value.toString();}
}
```

שאלה 1, בגרות קיץ תשס"ג 2003

חיה עידן

הניסוח המקורי

כתוב תת-תכנית (פרוצדורה או פונקציה) בסביבת העבודה, שתקבל רשימה L לא ריקה של מספרים שלמים ותחזיר רשימה חדשה באופן הזה:

עבור כל תת-רשימה של מספרים עולים ב-L, שבה כל מספר גדול מקודמו, יופיע ברשימה החדשה מספר אחד שהוא סכום כל המספרים העולים. כל תת-רשימה של מספרים עולים מסתיימת כאשר אחריה יש מספר שהוא קטן מן המספר האחרון שבה, או שווה לו.

תת-רשימה יכולה לכלול גם מספר אחד בלבד. סדר האיברים ברשימה החדשה יהיה על פי סדר התת-רשימות ברשימה L.

לדוגמה:

עבור הרשימה L (משמאל לימין): 7, 2, 4, 8, 20, 18, 19, 20, 20, 5, -3, 0, 9
הרשימה החדשה שתוחזר: 7, 34, 57, 20, 5, 6
הערה: אין צורך לממש בסביבת העבודה את הפעולות של ממשק רשימה.

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

כתוב תת-תכנית בסביבת העבודה, שתקבל שרשרת חוליות לא ריקה L של מספרים שלמים, הפעולה תחזיר שרשרת חוליות חדשה באופן הבא:

עבור כל סדרה של חוליות של מספרים עולים ב-L, תופיע חוליה אחת המכילה את סכום כל המספרים העולים. כל סדרת חוליות של מספרים עולים מסתיימת כאשר אחריה יש מספר שהוא קטן או שווה למספר האחרון שבה. סדר החוליות בשרשרת החוליות החדשה יהיה על פי סדר תת הסדרות עולות בשרשרת החוליות המקורית.

פתרון

```
public static Node<int> MakeSumList(Node<int> lst)
{
    Node<int> sumLst, p, ps;
    p = lst;
    ps = sumLst = null;
    int sum = 0;
    while (p != null)
    {
        sum += p.GetInfo();
        if (p.GetNext() == null || p.GetInfo() >= p.GetNext().GetInfo())
        {
            if (ps == null)
                sumLst = ps = new Node<int>(sum);
            else
            {
                ps.SetNext(new Node<int>(sum));
                ps = ps.GetNext();
            }
            sum = 0;
        }
    }
}
```

```

    p=p.Next();
}
return sumLst;
}

```

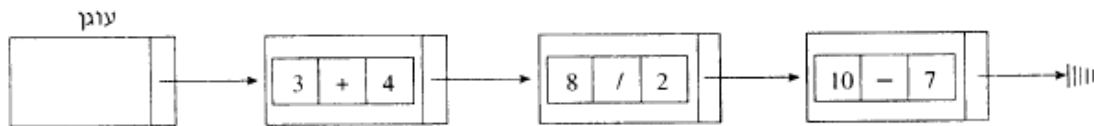
שאלה 2, בגרות קיץ תשס"ה 2005

הניסוח המקורי

רשימה "חשבונית" L היא רשימה שדה התוכן שלה מייצג ביטוי חשבוני. הביטוי החשבוני מורכב משלושה חלקים:

- מספר שלם גדול מאפס
- תו אחד מבין ארבעת התווים:
 - + המייצג חיבור
 - המייצג חיסור
 - * המייצג כפל
 - / המייצג חילוק
- מספר שלם גדול מאפס

דוגמה לרשימה "חשבונית" L:



- הגדר בסביבת העבודה את טיפוס שדה התוכן של איבר ברשימה "חשבונית" L.
- ממש בסביבת העבודה תת-תכנית calculate, שתקבל רשימה "חשבונית" L ומקום p ברשימה. p הוא מקום ב-L שאינו סוף-רשימה ואינו עוגן-רשימה. התת-תכנית תחזיר את התוצאה המתקבלת מהביטוי החשבוני הנמצא באיבר שבמקום p.
- ממש בסביבת העבודה תת-תכנית sumExpressions, שתקבל רשימה "חשבונית" L, ותחזיר את הסכום הכולל של תוצאות הביטויים החשבוניים הנמצאים ברשימה. בעבור רשימה ריקה יחזיר 0. עליך להשתמש בתת-תכנית calculate. בעבור הרשימה "החשבונית" L בדוגמה הנתונה, התת-תכנית sumExpressions תחזיר 14.

הצעה 1 של יבגני קנל

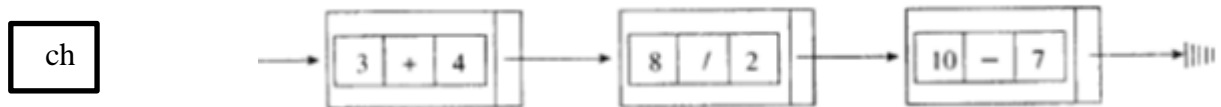
הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

שרשרת "חשבונית" היא שרשרת המורכבת מ"ביטויים חשבוניים". הביטוי החשבוני כולל שני מספרים שלמים ותו המציין אחת מ-5 פעולות חשבון: +, -, *, /. לפני תיאור חלקי של המחלקה MExp, המייצגת ביטוי החשבוני:

MExp	
private int num1;	המספר הראשון בביטוי
private char sign;	הסימן המגדיר פעולה חשבונית.

	הנחה: הסימן הוא אחד מ-5 סימנים : +, -, *, /, %.
<code>private int num2;</code>	המספר השני בביטוי
<code>public MExp(int num1, char sign, int num2)</code>	הפעולה הבונה
<code>public boolean isValid()</code>	הפעולה בודקת האם ניתן לבצע חישוב של ביטוי חשבוני
<code>public int calculate()</code>	הפעולה מחזירה את התוצאה המתקבלת מהביטוי החשבוני. אם לא ניתן לבצע חישוב, הפעולה תחזיר 0.

- א. ממש את פעולה `isValid()`
- ב. ממש את פעולה `calculate()`
- ג. ממש את פעולה החיצונית `sumExpressions(Node<MExp> ch)` המקבלת שרשרת "חשבונית". הפעולה תחזיר את הסכום הכולל של תוצאות הביטויים החשבוניים הנמצאים בשרשרת. בעבור השרשרת הבאה הפעולה תחזיר 14



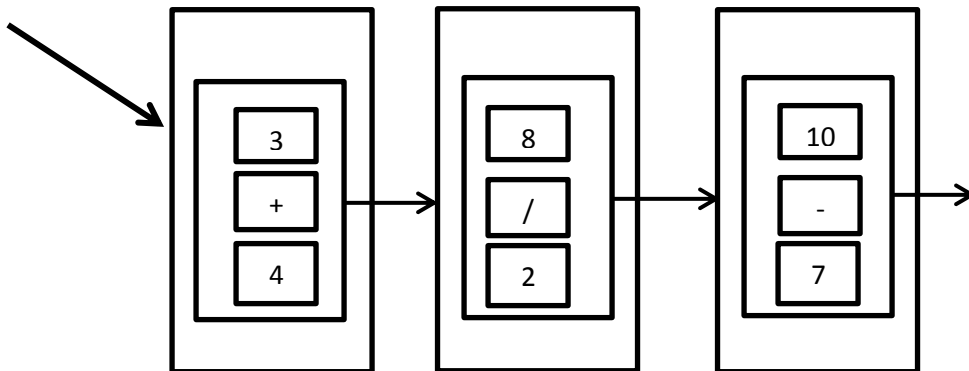
הצעה 2 של דורית ליקרמן

"רשימה חשבונית" היא רשימה של Expressions (ביטוי חשבוניים).

Expression (ביטוי חשבוני) מיוצג על ידי :

- מספר שלם גדול מאפס
- תו מבין ארבעת התווים: +
- מייצג חיבור
- מייצג חיסור
- * מייצג כפל
- / מייצג חילוק.
- מספר שלם גדול מאפס

דוגמא ל"רשימה חשבונית":



א. כתוב ב- c# או ב- Java את כותרת המחלקה **Expression** ואת תכונותיה.

ב. נתונה כותרת הפעולה: `calculate`:

```
public static int calculate (Node<Expression> p)
```

הפעולה מחזירה את ערך הביטוי החשבוני הנמצא ב- p.

ממש את הפעולה - c# או ב- Java.

ג. ממש ב- c# או ב- Java פעולה `sumExpressions`, המקבלת "רשימה חשבונית" ומחזירה את הסכום הכולל

של תוצאות הביטויים החשבוניים הנמצאים ברשימה

עליך להשתמש בפעולה **calculate**

לדוגמא, בעבוד הרשימה למעלה, `sumExpressions` תחזיר 14.

עבור סעיפים ב' ו- ג' הנח שכל פעולות ה- `get/set` במחלקה `Expression` קיימות.

שאלה 4, בגרות קיץ תשס"ו 2006

הניסוח המקורי:

כתוב תת-תכנית בפסקל או ב- C, שתקבל שתי רשימות L1, L2 של מספרים שלמים וגדולים מ- 0 ותחזיר רשימה חדשה L3.

התת-תכנית תסרוק את הרשימה L1 פעם אחת, מתחילתה עד סופה.

בכל שלב התת-תכנית תבדוק איבר אחד מהרשימה L1. נסמן את ערך האיבר ב- k.

התת-תכנית תבצע את אחת הפעולות שלהלן, בהתאם ל- k ולרשימה L2 כפי שהיא בשלב זה:

- אם k הוא מספר זוגי, התת-תכנית תמחק את האיבר שמיקומו k מתחילת הרשימה L2.
- אם k הוא מספר אי-זוגי, התת-תכנית תוסיף לרשימה L3 איבר שהערך שלו הוא הערך של האיבר שמיקומו k מתחילת הרשימה L2.
- אם ברשימה L2 אין איבר שמיקומו k, התת-תכנית לא תבצע דבר.

הערות:

אין חשיבות לסדר הכנסת האיברים לרשימה L3.

התת-תכנית לא תחזיר את הרשימה L2.

לדוגמא:

בעבור הרשימות L1, L2 (משמאל לימין):

L1 : 4, 3, 2, 6

L2 : 10, 11, 19, 1, 7, 100

מצב הרשימות L2, L3 בתום כל שלב של ביצוע התת-תכנית יהיה כזה:

I אחרי שלב I

L2 : 10, 11, 19, 7, 100

L3 :

II אחרי שלב II

L2 : 10, 11, 19, 7, 100

L3 : 19

אחרי שלב III

L2 : 10 , 19 , 7 , 100

L3 : 19

אחרי שלב IV

L2 : 10 , 19 , 7 , 100

L3 : 19

הצעה 1 ענת שלוש סגל

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

מצורפות בהמשך שתי הצעות פתרון: בתכנית הנוכחית ובתכנית החדשה.
ניתן לראות כי פתרון השאלה כפי שהיא בתכנית החדשה מעלה הופך להיות יותר מורכב.
מכיוון שיש צורך למצוא את האיבר ה-k ברשימה L2 ובהתאם לערכו אולי גם למחוק אותו יש לבצע סריקה שעוצרת באיבר ה-k-1.
בעיה נוספת שהייתה יכולה להיווצר היא מחיקת האיבר הראשון בשרשרת חוליות שהועברה כפרמטר לפעולה. אך בעיה זו לא קיימת מכיוון שמחיקה מתבצעת רק עבור איברים שמקומם ברשימה זוגי.
שתי הצעות ל"שיפוץ השאלה"

- עבור k זוגי הכנס ל L3 איבר שהערך שלו הוא ריבוע הערך של האיבר שמיקומו k מתחילת הרשימה L2.
(עבור k אי-זוגי אין שינוי)
- רק לבצע מחיקה מרשימה L2 עבור k זוגי, ולא לבצע דבר עבור k אי-זוגי.
כלומר אין בנייה והחזרה של רשימה שלישית.

הצעת פתרון לפי תכנית ישנה/נוכחית

```
public static List<Integer> getList ( List<Integer> lst1 , List<Integer> lst2)
{
    List<Integer> lst3 = new List<Integer>();
    Node<Integer> pos1= lst1.getFirst();
    while (pos1 != null)
    {
        int k = pos1.getInfo();
        int i = 1;
        Node<Integer> pos2 = lst2.getFirst();
        while (pos2 != null && i < k)
        {
            pos2 = pos2.getNext();
            i++;
        }
        if (i == k && pos2 != null)
        {
            if (k%2 == 0)
                lst2.remove(pos2);
        }
    }
}
```

```

        else
            lst3.insert (null, pos2.getInfo());
    }
    pos1 = pos1.getNext();
}
return lst3;
}

```

הצעת פתרון לפי התכנית החדשה

```

public static Node<Integer> getList ( Node<Integer> lst1 , Node<Integer> lst2)
{
    Node<Integer> lst3 = null;
    Node<Integer> pos1= lst1;
    while (pos1 != null)
    {
        int k = pos1. getValue ();
        int i = 1;
        Node<Integer> pos2 = lst2;
        while (pos2.getNext() != null && i < k-1)
        {
            pos2 = pos2.getNext();
            i++;
        }
        if (i == k-1 && pos2.getNext() != null)
        {
            if (k%2 == 0)
            {
                Node<Integer> temp = pos2.getNext();
                pos2.setNext(temp.getNext());
                temp.setNext(null);
            }
            else
                lst3=new Node<Integer> (pos2.getNext().getValue (), lst3);
        }
        pos1 = pos1.getNext();
    }
    return lst3;
}

```

הצעה 2 אוי גרינולד

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

- כתוב פעולה אשר מקבלת 2 רשימות של מספרים שלמים וחיוביים, list1, list2 . הפעולה עוברת על כל איברי רשימה list1 פעם אחת מתחילתה ועד סופה. עבור כל איבר מרשימה list1 אשר ערכו k , הפעולה תבצע את הצעד הבא:
 - אם k זוגי, הפעולה תמחק את האיבר שמיקומו k צעדים מתחילת רשימה List2
 - אם k אי-זוגי, הפעולה תוסיף לרשימה list3 את האיבר שנמצא במיקום k צעדים מתחילת רשימה list2
 - אם אין מיקום k ברשימה list2 , הפעולה לא תבצע כלום על איבר זה.
 - אין חשיבות לסדר הערכים ברשימה list3 .

list1: [2,4,1,5] list2: [10,20,30,40,50,60]	מצב התחלתי
k=2 list2: [10,30,40,50,60] list3: [0]	אחרי שלב 1
k=4 list2: [10,30,40,60] list3: [0]	אחרי שלב 2
k=1 list2: [10,30,40,60] list3: [10]	אחרי שלב 3
k=5 list2: [10,30,40,60] list3: [10]	אחרי שלב 4

- הפעולה תחזיר את רשימה list3 . במידה אף איבר לא התווסף לרשימה תוחזר רשימה עם חוליה אחת בלבד שערכה 0.
- לדוגמה עבור הרשימות:

פתרון:

```
import unit4.collectionsLib.Node;
import java.util.Scanner;
/**
 * @version 14/1/15
 * @author evi
 */
public class Bg4_06
{
    public static Node<Integer> create()
    { // פעולת ליצירת רשימה של שלמים
        Scanner in = new Scanner(System.in);
        int x;
        Node<Integer> first, node, pos;
        System.out.println("Enter integer value != 99");
        x = in.nextInt();
    }
}
```

```

first = new Node<Integer>(x);
pos = first;
System.out.println("Enter integer value, 99 to stop");
x = in.nextInt();
while (x!=99)
{
    node = new Node<Integer>(x);
    pos.setNext(node);
    pos = node;
    System.out.println("Enter integer value, 99 to stop");
    x = in.nextInt();
}
return first;
}

public static void print(Node<Integer> list)
{//עולה להדפסת רשימה של שלמים}
    while (list!=null)
    {
        System.out.print(list.getValue()+" --> ");
        list = list.getNext();
    }
    System.out.println("null");
}
/**
 * @param list: Node<Integer>
 * @param k: int
 * @return reference to Node<Integer> position k in list
 * if k>number of nodes in list, return null
 */
public static Node<Integer> goTo(Node<Integer> list, int k)
{
    while (list!=null && k>1)
    {
        list = list.getNext();
        k--;
    }
    return list;
}
/**
 * deletes node in k steps from beginning of list,
 * in conditions than there is k nodes
 * @param list: Node<Integer>
 * @param k ; int
 */
public static void deleteK(Node<Integer> list, int k)

```

```

    //k מקום איבר במקום k
    Node<Integer> prev = goTo(list, k-1);
    Node<Integer> pos;
    if (prev!=null && prev.hasNext())
    {
        pos = prev.getNext();
        prev.setNext(pos.getNext());
        pos.setNext(null);
    }
}
/**
 *
 * @param list2: Node<Integer>
 * @param list3: Node<Integer> refer to Node<Integer>(0)
 * @param k ; int
 * inserts values in k position of list2 to list3,
 * first value to insert change first value from 0 to the value from list2
 */
public static void insertK(Node<Integer> list2, Node<Integer> list3, int k)
{
    //list3 להשימה k במקום
    Node<Integer> pos2 = goTo(list2, k);
    Node<Integer> pos3 = list3;
    Node<Integer> node;
    int value;
    if (pos2!=null)
    {
        value = pos2.getValue();
        if (pos3.getInfo()==0)
        {
            // insert the first value
            pos3.setValue(value);
        }
        else
        {
            node = new Node<Integer>(value,pos3.getNext());
            pos3.setNext(node);
        }
    }
}
public static Node<Integer> doIt(Node<Integer> list1, Node<Integer> list2)
{
    // הפעולה על פי הניסוח
    int k;
    int value;
    Node<Integer> pos;
    Node<Integer> list3 = new Node<Integer>(0);
    while (list1!=null)
    {

```

```

k = list1.getValue();
if (k%2==0)
    // removes value after (k-1) steps in list2
    deleteK(list2, k);
}
else
    // k%2!=0
    insertK(list2, list3, k);
}
list1=list1.getNext();
}
return list3;
}
public static void main(String[] args)
{
    System.out.println("create list1:");
    Node<Integer> list1 = UtilNodeInt.create();
    UtilNodeInt.print(list1);
    System.out.println("create list2:");
    Node<Integer> list2 = UtilNodeInt.create();
    UtilNodeInt.print(list2);
    Node<Integer> list3 = doIt(list1, list2);
    System.out.println("after doIt");
    System.out.println("list2");
    UtilNodeInt.print(list2);
    System.out.println("list3");
    UtilNodeInt.print(list3);
}
}

```

הצעה 3 רחל בן-עמי ומושית ולץ

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

כתוב פעולה שתקבל שתי רשימות L1, L2 של מספרים שלמים וגדולים מ-0 ותחזיר רשימה חדשה L3.

L1, L2, L3 מטיפוס Node<int>.

הפעולה תסרוק את הרשימה L1 פעם אחת מתחילתה ועד סופה.

בכל שלב הפעולה תבדוק אבר אחד מהרשימה L1. נסמן את ערך האיבר ב-k.

הפעולה תבצע את אחת הפעולות שלהלן, בהתאם ל k ולרשימה L2 כפי שהיא בשלב זה:

- אם K הוא מספר זוגי, הפעולה תמחק את האיבר שמיקומו k מתחילת הרשימה L2.

- אם k הוא מספר אי זוגי, הפעולה תוסיף לרשימה L3 איבר שהערך שלו הוא הערך של האיבר

שמיקומו k מתחילת הרשימה L2.

- אם ברשימה L2 אין איבר שמיקומו k הפעולה לא תבצע דבר.

הערות:

אין חשיבות לסדר הכנסת האיברים לרשימה L3.
הפעולה לא תחזיר את הרשימה L2.
לדוגמה:

עבור הרשימות L1, L2 (משמאל לימין):

L1 → 4, 3, 2, 6

L2 → 10, 11, 19, 1, 7, 100

מצב הרשימות L2, L3 בתום כל שלב:

שלב 1: L2 → 10, 11, 19, 7, 100

L3 →

שלב 2: L2 → 10, 11, 19, 7, 100

L3 → 19

שלב 3: L2 → 10, 19, 7, 100

L3 → 19

שלב 4: L2 → 10, 19, 7, 100

L3 → 19

שאלה 4, בגרות קיץ תשס"ז 2007

הניסוח המקורי

בטלפון הנייד של חברת "נייד-פון" ניהול השיחות הנכנסות מתבצע באופן הזה:

- מספר טלפון שממנו התקבלה שיחה נשמר ברשימת שיחות נכנסות.
- מספרי הטלפון של השיחות הנכנסות שהתקבלו במכשיר נשמרים לפי סדר הפוך מסדר קבלתן, כך שמספר הטלפון של השיחה האחרונה שהתקבלה יהיה ראשון ברשימת השיחות הנכנסות.
- כל מספרי הטלפון השמורים שונים זה מזה. כאשר מתקבלת שיחה נוספת ממספר טלפון הנמצא כבר בין מספרי השיחות הנכנסות, מספר זה נרשם ראשון והוא נמחק מקומו הקודם.
- מספר השיחות הנכנסות שאפשר לשמור במכשיר מוגבל.
- כאשר רשימת השיחות הנכנסות מלאה ומתקבלת שיחה ממספר טלפון שאינו נמצא ברשימה, מספר זה נרשם ראשון והמספר האחרון נמחק.

יומן שיחות נכנסות מיוצג על ידי:

1. רשימה מקושרת שכל איבר בה הוא מספר טלפון המיוצג ע"י מחרוזת.
2. מספר שלם maxCalls המייצג את המספר המקסימלי של מספרי טלפון שאפשר לשמור ביומן השיחות הנכנסות.
3. מספר שלם currentCalls המייצג את המספר של מספרי הטלפון הנמצאים בזמן מסוים ביומן השיחות הנכנסות.

לפותרים ב- C# או ב- Java

א. כתוב ב- C# או ב- Java את כותרת המחלקה יומן שיחות נכנסות, ואת התכונות שלה על פי הייצוג המתואר ב- 1 עד 3

ב. ממש ב- C# או ב- Java פעולה במחלקה יומן שיחות נכנסות, המקבלת מספר טלפון של שיחה נכנסת tel ומעדכנת את יומן השיחות הנכנסות.

הערה: אפשר להשתמש בפעולות הממשק $List < T >$ בלי לממש אותן. אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

הצעה 1 רוני אלנקרי

הקושי אליו אני מתייחסת הוא למחיקה מרשימה שהופכת מורכבת יותר בשרשרת חוליות שכן יש לשמור את המקום הקודם לחוליה שיש למחוק. בנוסף קיים הקושי שיש להתייחס ל-2 מקרים שונים, מחיקה של החוליה הראשונה ומחיקה של חוליה שאינה הראשונה.

במטרה להוריד את רמת הקושי ניתן לנסח את השאלה של בגרות 2007 שאלה 4 כך שלא תתקיים מחיקה מהמקום הראשון, זאת מבלי לשנות את הנדרש בתיאור.

הצעה לניסוח חדש שייתאים לתכנית "מבני נתונים":

בטלפון הנייד של חברת "נייד-פון" ניהול השיחות הנכנסות מתבצע באופן הזה:

- מספר טלפון שממנו התקבלה שיחה נשמר ברשימת שיחות נכנסות.
- מספרי הטלפון של השיחות הנכנסות שהתקבלו במכשיר נשמרים לפי סדר הפוך מסדר קבלתן, כך שמספר הטלפון של השיחה האחרונה שהתקבלה יהיה ראשון ברשימת השיחות הנכנסות.
- כל מספרי הטלפון השמורים שונים זה מזה. כאשר מתקבלת שיחה נוספת ממספר טלפון הנמצא כבר בין מספרי השיחות הנכנסות, מספר זה נרשם ראשון והוא נמחק מקומו הקודם. אם המספר נמצא כבר בין מספרי

הטלפון והוא ראשון אין לעשות דבר.

- מספר השיחות הנכנסות שאפשר לשמור במכשיר מוגבל.
- כאשר רשימת השיחות הנכנסות מלאה ומתקבלת שיחה ממספר טלפון שאינו נמצא ברשימה, מספר זה נרשם ראשון והמספר האחרון נמחק.

הצעה 2 יוסי זהבי

הצעה לניסוח חדש שייתאים לתכנית "מבני נתונים":

בטלפון הנייד של חברת "נייד-פון" ניהול השיחות הנכנסות מתבצע באופן הזה:

- מספר טלפון שממנו התקבלה שיחה נשמר ברשימת שיחות נכנסות.

- מספרי הטלפון של השיחות הנכנסות שהתקבלו במכשיר נשמרים לפי סדר הפוך מסדר קבלתן, כך שמספר הטלפון של השיחה האחרונה שהתקבלה יהיה ראשון ברשימת השיחות הנכנסות.
- כל מספרי הטלפון השמורים שונים זה מזה. כאשר מתקבלת שיחה נוספת ממספר טלפון הנמצא כבר בין מספרי השיחות הנכנסות, מספר זה נרשם ראשון והוא נמחק מקומו הקודם.
- מספר השיחות הנכנסות שאפשר לשמור במכשיר מוגבל.
- כאשר רשימת השיחות הנכנסות מלאה ומתקבלת שיחה ממספר טלפון שאינו נמצא ברשימה, מספר זה נרשם ראשון והמספר האחרון נמחק.

יומן שיחות נכנסות מיוצג על ידי:

1. שרשרת חוליות שכל חוליה בה הוא מספר טלפון המיוצג ע"י מחרוזת.
2. מספר שלם maxCalls המייצג את המספר המקסימלי של מספרי טלפון שאפשר לשמור ביומן השיחות הנכנסות.
3. מספר שלם currentCalls המייצג את המספר של מספרי הטלפון הנמצאים בזמן מסוים ביומן השיחות הנכנסות.

לפותרים ב-C# או ב-Java

- א. כתוב ב-C# או ב-Java את כותרת המחלקה יומן שיחות נכנסות, ואת התכונות שלה על פי הייצוג המתואר ב- 1 עד 3
- ב. ממש ב-C# או ב-Java פעולה במחלקה יומן שיחות נכנסות, המקבלת מספר טלפון של שיחה נכנסת tel ומעדכנת את יומן השיחות הנכנסות.

הערה: אפשר להשתמש בפעולות של המחלקה `Node<T>`, בלי לממש אותן.

אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

שאלה 2, בגרות קיץ תשס"ח 2008

הניסוח המקורי

טיפוס הנתונים **מיון** - בתחומים מאפשר לשמור כמות גדולה של מספרים שלמים, שונים זה מזה, הממוינים לפי תחומים. לכל תחום מוגדר ערך מינימלי וערך מקסימלי. כל תחום כולל מספרים הגדולים מהערך המינימלי או שווים לו, וקטנים מהערך המקסימלי.

לתחום האחרון מוגדר רק ערך מינימלי, והוא כולל מספרים הגדולים מערך או שווים לו.

טיפוס הנתונים **מיון** - בתחומים מורכב מ:

mins מערך חד מימדי בגודל 100 המכיל מספרים שלמים שונים זה מזה, הממוינים בסדר עולה. כל מספר מייצג ערך מינימלי של תחום.

values מערך חד מימדי בגודל 100 המכיל רשימות כל רשימה מכילה מספרים שלמים שונים זה מזה, הממוינים בסדר עולה. ברשימה שנמצאת במקום k במערך values נמצאים מספרים הגדולים מן הערך שנמצא במקום k במערך mins או שווים לו. וקטנים מן הערך שנמצא במקום k+1 במערך mins. ברשימה האחרונה נמצאים מספרים הגדולים מן הערך המינימלי של התחום האחרון או שווים לו.

current מספר שלם המייצג את מספר התחומים הנוכחי במערך mins

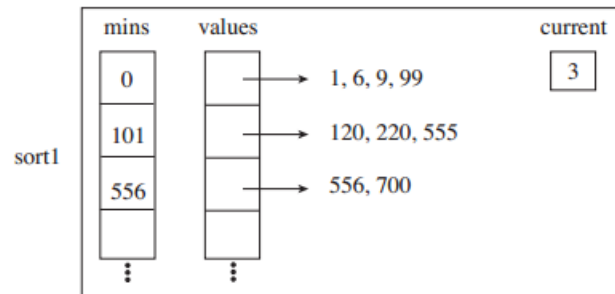
לדוגמא: בעבור המספרים האלה: 1, 700, 556, 120, 220, 9, 6, 99, 555 ושלושת התחומים האלה:

i. מספרים הגדולים מ 0 או שווים לו וקטנים מ 101

ii. מספרים הגדולים מ 101 או שווים לו וקטנים מ 556

iii. מספרים השווים ל 556 או גדולים ממנו.

sort1 מטיפוס **מיון** – בתחומים יהיה:



- א. רשום ב Java או ב C# את כותרת המחלקה **מיון** – בתחומים **SortByRange** ואת התכונות שלה.
- ב. ממש ב Java או ב C# פעולה פנימית בשם insert במחלקה **SortByRange**, שתקבל מספר שלם num. הפעולה תכניס את num לרשימה המתאימה במערך values, על פי סדר המיון של המספרים ברשימה. הנח כי num אינו נמצא ב- values ו- num גדול מהערך הקטן ביותר שנמצא במערך mins.
- ג. ממש ב Java או ב C# פעולה פנימית בשם addRange במחלקה **SortByRange**, שתקבל מספר שלם r הגדול מכל המספרים ב- values. הפעולה תוסיף תחום חדש ש- r הוא הערך המינימלי שלו. הנח כי יש מקום לתחום נוסף.

ד. 1. מהי סיבוכיות זמן ריצה של הפעולה שמימשת בסעיף ב'. הסבר.

2. מהי סיבוכיות זמן ריצה של הפעולה שמימשת בסעיף ג'. הסבר.

הערה: אפשר להשתמש בפעולות של המחלקה **Node<T> List<T>**, בלי לממש אותן.

אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

הצעה 1 רחל לודמר

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

טיפוס הנתונים מיון - בתחומים מאפשר לשמור כמות גדולה של מספרים שלמים, שונים זה מזה, הממוינים לפי תחומים. לכל תחום מוגדר ערך מינימלי וערך מקסימלי. כל תחום כולל מספרים הגדולים מהערך המינימלי או שווים לו, וקטנים מהערך המקסימלי.

לתחום האחרון מוגדר רק ערך מינימלי, והוא כולל מספרים הגדולים מערך או שווים לו.

טיפוס הנתונים מיון - בתחומים מורכב מ:

mins – מערך חד מימדי בגודל 100 המכיל מספרים שלמים שונים זה מזה, הממוינים בסדר עולה. כל מספר מייצג ערך מינימלי של תחום.

values - מערך חד מימדי בגודל 100 המכיל רשימות כל רשימה מכילה מספרים שלמים שונים זה מזה, הממוינים בסדר עולה.

הרשימות הן מטיפוס Node<Integer> ב Java, וב C# מטיפוס Node<int>.

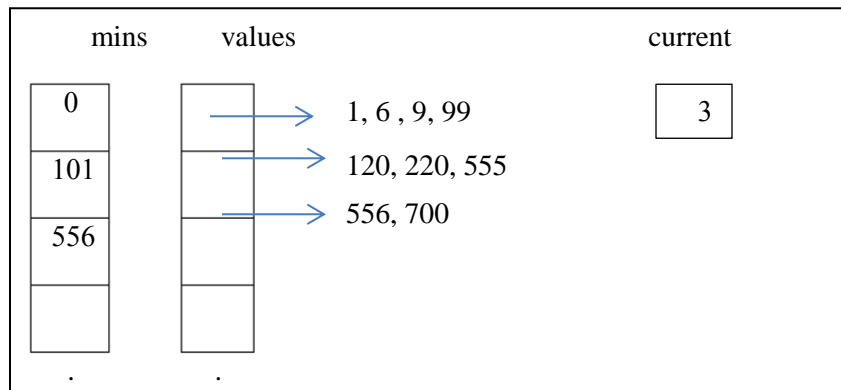
ברשימה שנמצאת במקום k במערך values נמצאים מספרים הגדולים מן הערך שנמצא במקום k במערך mins או שווים לו. וקטנים מן הערך שנמצא במקום k+1 במערך mins. ברשימה האחרונה נמצאים מספרים הגדולים מן הערך המינימלי של התחום האחרון או שווים לו.

current – מספר שלם המייצג את מספר התחומים הנוכחי במערך mins.

לדוגמא: בעבור המספרים האלה: 1, 700, 556, 120, 220, 9, 6, 99, 555
ושלושת התחומים האלה:

- .iv מספרים הגדולים מ 0 או שווים לו וקטנים מ 101
- .v מספרים הגדולים מ 101 או שווים לו וקטנים מ 556
- .vi מספרים השווים ל 556 או גדולים ממנו.

sort1 מטיפוס מיון – בתחומים יהיה:



א. רשום ב Java או ב C# את כותרת המחלקה מיון – בתחומים SortByRange ואת התכונות שלה.

הנח שבמחלקה פעולה בונה ברירת מחדל ופעולה שמקבלת פרמטר לכל תכונה.

ב. ממש ב Java או ב C# פעולה פנימית בשם insert במחלקה SortByRange, שתקבל מספר שלם num.

הפעולה תכניס את num לרשימה המתאימה במערך values, על פי סדר המיון של המספרים ברשימה.

הנח כי num אינו נמצא ב-values ו-num גדול מהערך הקטן ביותר שנמצא במערך mins.

ג. ממש ב Java או ב C# פעולה פנימית בשם `addRange` במחלקה `SortByRange`, שתקבל מספר שלם `r` הגדול מכל המספרים ב- `values`. הפעולה תוסיף תחום חדש ש- `r` הוא הערך המינימלי שלו. הנח כי יש מקום לתחום נוסף.

- ד.1. מהי סיבוכיות זמן ריצה של הפעולה שמימשת בסעיף ב'. הסבר.
 - ד.2. מהי סיבוכיות זמן ריצה של הפעולה שמימשת בסעיף ג'. הסבר.
- הערה: אפשר להשתמש בפעולות של המחלקה `Node<T>`, בלי לממש אותן. אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

הצעה 2 גיא אורן

למה בחרתי בשאלה הזו. לדעתי השאלה הזו מייצגת את רמת השאלות שהייתי רוצה לראות בבגרות.

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

מבנה הנתונים **מיון בתחומים** (`SortByRange`) מאפשר לשמור כמות גדולה של מספרים **שלמים**, שונים זה מזה, הממוינים לפי תחומים. כל המספרים **בתחום** נעים בין ערך מינימלי למקסימלי המגדירים את התחום. לתחום האחרון יוגדר ערך מינימלי בלבד.

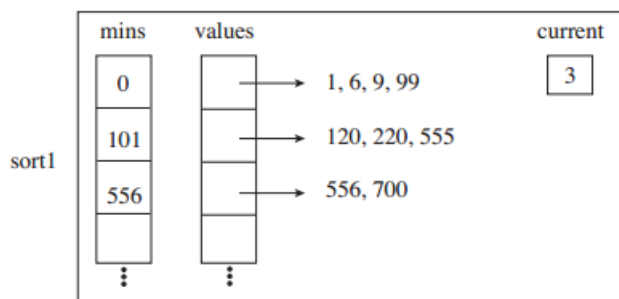
כדי לממש את מבנה הנתונים נגדיר:

`mins` מערך חד ממדי בגודל 100 המכיל מספרים בסדר עולה ממש.

`values` מערך חד ממדי בגודל 100 המכיל הפנייה לשרשרת חוליות **ממוינת**. שרשרת חוליות `k` מכילה מספרים המתאימה לתחום המוגדר ע"י `(mins[k], mins[k+1])`.
`current` – מספר המייצג את מספר התחומים הנוכחי.

לדוגמה: בעבור המספרים האלה: 1, 700, 556, 120, 220, 9, 6, 99, 555

`sort1` הוא אובייקט/הפנייה מסוג `SortByRanger`



- א. רשום ב JAVA את כותרת המחלקה `SortByRange` ואת התכונות שלה.
- ב. ממש את הפעולה `insert`¹ כך שתקבל מספר `num`. הפעולה תכניס את המספר לשרשרת החוליות המתאימה. ניתן להניח כי `num` גדול מהערך המינימלי במערך התחומים². ניתן להניח שהמספר `num` אינו קיים במבנה.
- ג. ממש ב JAVA פעולה בשם `addRange` שתקבל מספר `r` הגדול ממש מהמספר המקסימלי ב- `values`. הפעולה תוסיף תחום חדש ש- `r` הוא ערכו המינימלי. הנח כי יש מקום לתחום הנוסף³.

¹ אני לא חושב שצריך לציין פנימית, זו צריכה להיות ברירת המחדל.

² עדיף היה לו ההנחה הזו הייתה יורדת, ובמקומה הצעה מה לעשות במקרה הנ"ל.

³ שוב מיותר. במקום יש להגדיר לתלמידים מה לעשות במידה ואין אפשרות להוסיף תחום נוסף.

אני חושב שהיה עדיף להפוך את סדר הסעיפים. ב וג'.

פתרון

התלבטתי אם להוסיף sortedList משלי. החלטתי בסוף ללכת על פתרון פשוט יותר.

```
package bagrut2008;
import unit4.collectionsLib.*;
public class SortByRange {

    final static int MAX_NUM_OF_RANGES = 100;
    private Node<Integer>[] values = new Node[MAX_NUM_OF_RANGES];
    private int[] mins = new int[MAX_NUM_OF_RANGES];
    private int current;

    private int search(int num)
    {
        for(int idx = 0; idx <= current; idx++)
        {
            if(mins[idx] < num)
                return idx;
        }
        return -1;
    }

    public void addRange(int r)
    {
        mins[current++] = r;
    }

    public void insert(int num)
    {
        Node<Integer> cur = values[search(num)],
                    prev = cur;
        Node<Integer> newLink = new Node<Integer>(num);
        if(cur == null)
        {
            cur = newLink;
        }
        else
        {
            for(; cur.getInfo() < num; prev = cur, cur = cur.getNext());
            prev.setNext(newLink);
            newLink.setNext(cur);
        }
    }
}
```

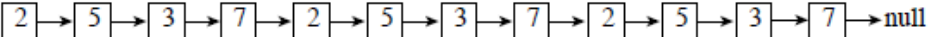
שאלה 1, בגרות קיץ תשס"ט 2009

הניסוח המקורי:

רשימה תיקרא **משולשת** אם היא מקיימת את התנאים האלה:

- * הרשימה אינה ריקה.
- * מספר האיברים בה מתחלק ב-3 בלי שארית.
- * האיברים בשליש הראשון של הרשימה מכילים את אותם ערכים שמכילים האיברים בשליש השני של הרשימה ואותם ערכים שמכילים האיברים בשליש השלישי של הרשימה. הערכים מסודרים באותו סדר בכל אחד מהשלישים.

לדוגמה: הרשימה L1 שלפניך היא רשימה **משולשת** באורך 12.

L1: 

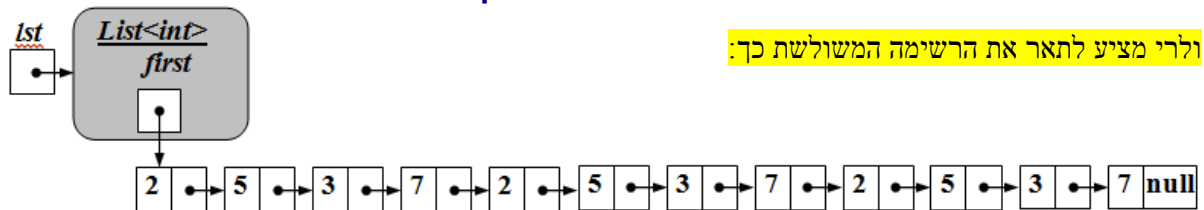
כתוב ב-C# או ב-Java פעולה חיצונית שתקבל רשימה L שהאיברים שלה הם מטיפוס שלם.

אם L היא רשימה **משולשת**, הפעולה תחזיר true. אחרת - הפעולה תחזיר false.

אתה יכול להשתמש בפעולות של המחלקות `Node <T>`, `List <T>` בלי לממש אותן.

אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

הצעה 1 ולרי פקר



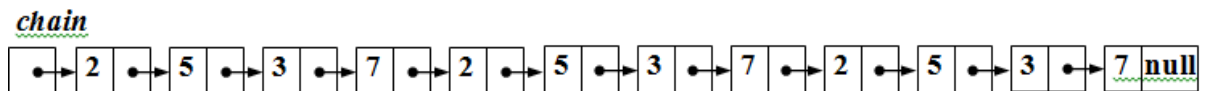
ולרי מציע לתאר את הרשימה המשולשת כך:

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

שרשרות חוליות תיקרא **משולשת** אם היא מקיימת את התנאים האלה:

- * מספר האיברים בה מתחלק ב-3 בלי שארית.
- * האיברים בשליש הראשון של שרשרות החוליות מכילים את אותם ערכים שמכילים האיברים בשליש השני של שרשרות החוליות ואותם ערכים שמכילים האיברים בשליש השלישי של שרשרות החוליות. הערכים מסודרים באותו סדר בכל אחד מהשלישים.

לדוגמה: שרשרות החוליות chain שלפניך היא רשימה של מספרים שלמים **משולשת** באורך 12.



כתוב פעולה חיצונית שתקבל שרשרות חוליות chain שהאיברים שלה הם מטיפוס שלם.

אם chain היא שרשרות חוליות משולשת, הפעולה תחזיר true. אחרת - הפעולה תחזיר false.

הצעה 2 דפנה לוי רשתי

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

שרשרת חוליות L תקרא משולשת אם היא מקיימת את התנאים הבאים:
מספר החוליות בה מתחלק ב-3 בלי שארית.

החוליות בשליש הראשון של השרשרת מכילים את אותם ערכים שמכילות החוליות בשליש השני של השרשרת ואותם ערכים שמכילות החוליות בשליש השלישי של השרשרת. הערכים מסודרים באותו סדר בכל אחד מהשלישים.

כתוב פעולה שתקבל שרשרת חוליות L שהאיברים שלה הם מטיפוס שלם. אם L היא שרשרת חוליות משולשת הפעולה תחזיר אמת, אחרת, הפעולה תחזיר שקר.

```
public static bool TripleList(Node<int> pos1)
{
    int length = Length(pos1);
    if (length == 0 || length % 3 != 0)
        return false;
    Node<int> pos2 = NodeAfterXNodes(pos1, length / 3);
    Node<int> pos3 = NodeAfterXNodes(pos2, length / 3);

    while (pos3 != null &&
           pos1.GetInfo() == pos2.GetInfo() &&
           pos1.GetInfo() == pos3.GetInfo())
    {
        pos1 = pos1.GetNext();
        pos2 = pos2.GetNext();
        pos3 = pos3.GetNext();
    }
    return (pos3 == null);
}
```

```
public static int Length(Node<int> pos)
{
    int length = 0;
    while (pos != null)
    {
        pos = pos.GetNext();
        length++;
    }
    return length;
}
```

```
private static Node<int> NodeAfterXNodes(Node<int> pos,
                                         int length)
{

```

```

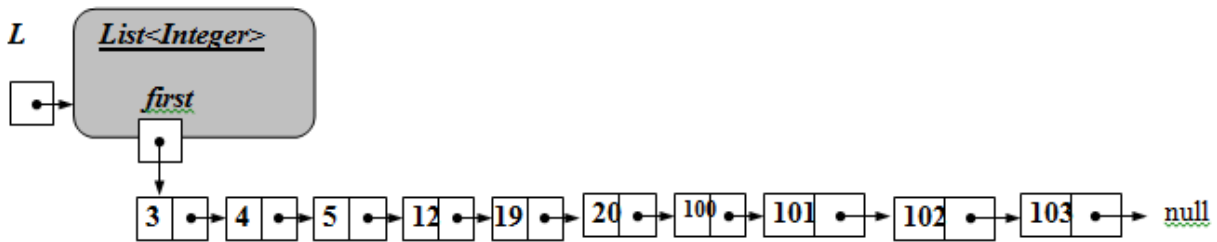
while (length > 0 && pos != null)
{
    pos = pos.getNext();
    length--;
}
if (length == 0) return pos;
return null;
}

```

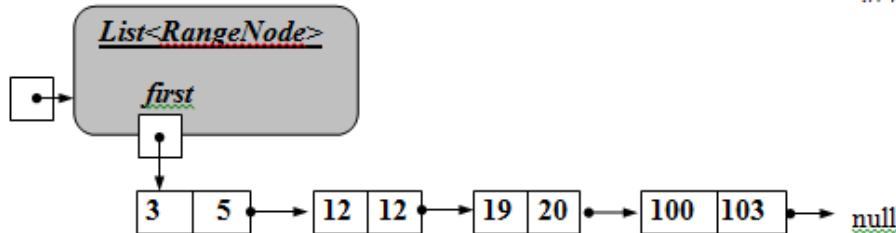
שאלה 2, בגרות קיץ תש"ע 2010

הניסוח המקורי:

L היא רשימה המכילה מספרים שלמים שונים זה מזה וממוינים בסדר עולה. רשימת הטוחים של L היא רשימה חדשה שנבנית באופן הזה: בעבור כל רצף של מספרים עוקבים ב-L יהיה ברשימת הטוחים איבר אחד שמכיל שני מספרים. מספר אחד הוא המספר הקטן ביותר ברצף, והמספר השני הוא הגדול ביותר ברצף. רצף יכול להיות באורך 1 או יותר. אם הרצף באורך 1, הוא מיוצג ברשימת הטוחים על ידי איבר ששני המספרים בו שווים.



רשימת הטוחים של L תהיה:



לפניך תיאור חלקי של המחלקה **RangeNode**, המייצגת איבר ברשימת הטוחים.

RangeNode	
private int from;	// המספר הקטן ביותר ברצף
Private int to	// המספר הגדול ביותר ברצף
public RangeNode(int from, int to)	

ממש ב- java או ב- c# פעולה חיצונית שתקבל רשימה לא ריקה, המכילה מספרים שלמים השונים זה מזה וממוינים בסדר עולה, ותחזיר את רשימת הטוחים שלה.

כותרת הפעולה ב- java היא:

```
public static List<RangeNode> createRangList(List<Integer> sourceList)
```


כותרת הפעולה ב- c# היא:

```
public static List<RangeNode> CreateRangList(List<int> sourceList)
```

הנח שלכל אחת מהתכונות במחלקה **RangeNode** יש פעולות get -1 .set . אתה יכול להשתמש הפעולות אלה ובפעולה הבונה של המחלקה **RangeNode** בלי לממש אותן. אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

הצעה 1 גלית שריקי

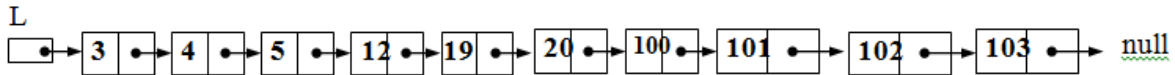
הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

L היא שרשרת חוליות המכילה מספרים שלמים שונים זה מזה וממוינים בסדר עולה.

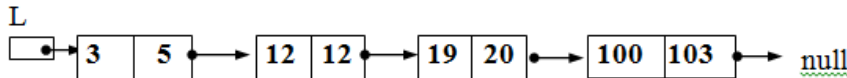
רשימת הטווחים של L היא שרשרת חוליות חדשה שנבנית באופן הזה:

בעבור כל רצף של מספרים עוקבים ב-L יהיה ברשימת הטווחים איבר אחד שמכיל שני מספרים. מספר אחד הוא המספר הקטן ביותר ברצף, והמספר השני הוא הגדול ביותר ברצף.

רצף יכול להיות באורך 1 או יותר. אם הרצף באורך 1, הוא מיוצג ברשימת הטווחים על ידי איבר ששני המספרים בו שווים.



רשימת הטווחים של L תהיה:



לפניך תיאור חלקי של המחלקה **RangeNode**, המייצגת איבר ברשימת הטווחים.

RangeNode	
private int from;	// המספר הקטן ביותר ברצף
Private int to	// המספר הגדול ביותר ברצף
public RangeNode(int from, int to)	

ממש ב- java או ב- c# פעולה חיצונית שתקבל שרשרת חוליות, המכילה מספרים שלמים השונים זה מזה וממוינים בסדר עולה, ותחזיר את רשימת הטווחים שלה.

כותרת הפעולה ב- java היא:

```
public static Node<RangeNode> createRangList(Node<Integer> sourceList)
```

כותרת הפעולה ב- c# היא:

```
public static Node<RangeNode> CreateRangList(Node<int> sourceList)
```

הנח שלכל אחת מהתכונות במחלקה **RangeNode** יש פעולות get -1 .set . אתה יכול להשתמש הפעולות אלה ובפעולה הבונה של המחלקה **RangeNode** בלי לממש אותן. אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

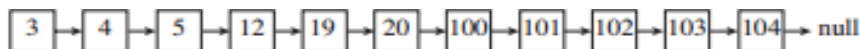
הערה של דפנה מינסטר:

בתרגיל שהמרת קיים המושג שרשרת חוליות, (L היא שרשרת חוליות המכילה), אך בהמשך השאלה יש את המונח איבר. אני חושבת שמונח זה קשור יותר לרשימה ולא לשרשרת חוליות. אולי גם כדאי לשנות את רשימת הטווחים לשרשרת הטווחים? הייתי מחליפה את המילה איבר במילה חוליה. מה דעתך? מה דעתכם, חברי המורים?

הצעה 2 חני טוראל

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

intNode הוא הפנייה לחוליה הראשונה בשרשרת חוליות המכילה מספרים שלמים השונים זה מזה וממיינים בסדר עולה – לשרשרת זו נקרא רשימה L. **רשימת הטווחים** של הרשימה L היא שרשרת חוליות חדשה שנבנית באופן הזה: בעבור כל רצף של מספרים עוקבים ברשימה L יהיה ברשימת הטווחים חוליה אחת שמכילה שני מספרים. מספר אחד הוא המספר הקטן ביותר ברצף, והמספר השני הוא המספר הגדול ביותר ברצף. רצף יכול להיות באורך 1 או יותר. אם הרצף הוא באורך 1, הוא מיוצג ברשימת הטווחים על ידי חוליה ששני המספרים בה שווים. לדוגמא, בעבור הרשימה L שלפניך:



רשימת הטווחים של L תהיה:



לפניך תיאור חלקי של המחלקה **RangeNode**, המייצגת חוליה ברשימת הטווחים.

RangeNode	
private int from;	// המספר הקטן ביותר ברצף
private int to;	// המספר הגדול ביותר ברצף
public RangeNode(int from, int to)	

ממש ב Java או ב C# פעולה חיצונית שתקבל הפנייה לשרשרת חוליות, המכילה מספרים שלמים שונים זה מזה וממיינים בסדר עולה, ותחזיר את רשימת הטווחים שלה (את הפניה לחוליה הראשונה בשרשרת החוליות).

כותרת הפעולה ב- Java היא:

```
public static Node<RangeNode> createRangeList(Node<int> source)
```

כותרת הפעולה ב- C# היא:

```
public static Node<RangeNode> CreateRangeList(Node<int> source)
```

הנח שלכל אחת מהתכונות במחלקה **RangeNode** יש פעולות get ו-set .

אתה יכול להשתמש בפעולות אלה ובפעולה הבונה של המחלקה **RangeNode** בלי לממש אותן.

אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

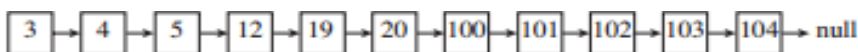
הצעה 3 ישראל אברמוביץ

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

L היא רשימה אשר ממומשת כשרשרת חוליות המכילה מספרים שלמים שונים זה מזה וממוינים בסדר עולה. רשימת הטווחים של L היא רשימה חדשה שנבנית באופן הזה: בעבור כל רצף של מספרים עוקבים ב-L יהיה ברשימת הטווחים איבר אחד שמכיל שני מספרים. מספר אחד הוא המספר הקטן ביותר ברצף, והמספר השני הוא המספר הגדול ביותר ברצף.

רצף יכול להיות באורך 1 או יותר. אם הרצף הוא באורך 1, הוא מיוצג ברשימת הטווחים על ידי חוליה ששני המספרים בה שווים.

לדוגמא, בעבור הרשימה L שלפניך:



רשימת הטווחים של L תהיה:



לפניך תיאור חלקי של המחלקה **RangeNode**, המייצגת חוליה ברשימת הטווחים.

RangeNode	
private int from;	// המספר הקטן ביותר ברצף
private int to;	// המספר הגדול ביותר ברצף
public RangeNode(int from, int to)	

ממש ב Java או ב C# פעולה חיצונית שתקבל הפניה אל החוליה הראשונה ברשימה, הרשימה המכילה לפחות חוליה אחת. שרשרת החוליות שמרכיבה את הרשימה, מכילה מספרים שלמים שונים זה מזה וממוינים בסדר עולה.

הפעולה תחזיר הפניה לרשימה שבנויה משרשרת החוליות **RangeNode**.

כותרת הפעולה ב Java

```
public static Node<RangeNode> CreateRangeListWithNodes(Node<integer> sourceList)
```

כותרת הפעולה ב C#

```
public static Node<RangeNode> CreateRangeListWithNodes(Node<int> sourceList)
```

הנח שלכל אחת מהתכונות במחלקה **RangeNode** יש פעולות get ו-set.

אתה יכול להשתמש בפעולות אלה ובפעולה הבונה של המחלקה **RangeNode** בלי לממש אותן.

אם אתה משתמש בפעולות נוספות, עליך לממש אותן.

פיתרון (שינוי מהפתרון באתר של הילה קדמן)

```

public static Node<RangeNode> CreateRangeListWithNodes(Node<int> sourceList)
{
    Node<int> p = sourceList;
    Node<RangeNode> qFirst = new Node<RangeNode>(null, null);

```

```

Node<RangeNode> q = qFirst;
Node<RangeNode> qTemp;

RangeNode r = new RangeNode(2,3);
Node<RangeNode> qFirst2 = new Node<RangeNode>(r, null);

int x, y = 0, z;

z = p.GetInfo();
if (p.GetNext() == null) // אחד מאיבר מורכבת הרשימה אם
    qFirst.SetInfo(new RangeNode(z, z));
else
{
    while (p.GetNext() != null)
    {
        x = p.GetInfo();
        y = p.GetNext().GetInfo();
        if (y - x == 1)
        {
            p = p.GetNext();
        }
        else
        {
            qTemp = new Node<RangeNode>(new RangeNode(z, x), null);
            q.SetNext(qTemp);
            q = q.GetNext();
            z = y;
            p = p.GetNext();
        }
    }
    qTemp = new Node<RangeNode>(new RangeNode(z, y), null);
    q.SetNext(qTemp);
}
return qFirst;
}

```

שאלה 1, בגרות קיץ תשע"ג 2013

הניסוח המקורי:

לפניך שלוש פעולות Sod1, Sod2, Sod3.

```

public static bool Sod1(List<int> list1, List<int> list2)
{
    Node<int> node1 = list1.GetFirst();
    Node<int> node2 = list2.GetFirst();
    for (int i = 1; i <= 4; i++)
    {
        if (node1 == null || node2 == null)

```

```

        return false;
    if (i == 1 || i == 4)
        if (node1.GetInfo() != node2.GetInfo())
            return false;
        node1 = node1.GetNext();
    }
    return true;
}
public static bool Sod2(List<int> list1, List<int> list2)
{
    Node<int> node1 = list1.GetFirst();
    Node<int> node2 = list2.GetFirst();
    while (node1 != null && node2 != null)
    {
        if (node1.GetInfo() != node2.GetInfo())
            return false;
        node1 = node1.GetNext();
        node2 = node2.GetNext();
    }
    return true;
}
public static bool Sod3(List<int> list1, List<int> list2)
{
    Node<int> node1 = list1.GetFirst();
    while (node1 != null)
    {
        bool found = false;
        Node<int> node2 = list2.GetFirst();
        while (node2 != null && !found)
        {
            if (node1.GetInfo() == node2.GetInfo())
                found = true;
            node2 = node2.GetNext();
        }
        if (!found)
            return false;
        node1 = node1.GetNext();
    }
    return true;
}

```

א. עקוב אחר כל אחת מהפעולות Sod1, Sod2, Sod3, עם הרשימות list1 ו-list2 שלפניך:

list1: [2, 4, 5, 1, 1, 9]

list2: [2, 4, 5, 1, 4]

רשום מה יוחזר בעבור כל אחת מהפעולות. במעקב הראה את המעבר על הרשימות.

ב. מה סיבוכיות זמן הריצה של כל אחת מהפעולות Sod1, Sod2, Sod3? נמק את תשובתך.

הצעה 1 ולרי פקר

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

לפניך שלוש פעולות Sod1, Sod2, Sod3.

```
public static bool Sod1(Node<int> chain1, Node<int> chain2)
{
    for (int i = 1; i <= 4; i++)
    {
        if (chain1 == null || chain2 == null)
            return false;
        if (i == 1 || i == 4)
            if (chain1.GetInfo() != chain2.GetInfo())
                return false;
        chain1 = chain1.GetNext();
    }
    return true;
}
```

```
public static bool Sod2(Node<int> chain1, Node<int> chain2)
{
    while (chain1 != null && chain2 != null)
    {
        if (chain1.GetInfo() != chain2.GetInfo())
            return false;
        chain1 = chain1.GetNext();
        chain2 = chain2.GetNext();
    }
    return true;
}
```

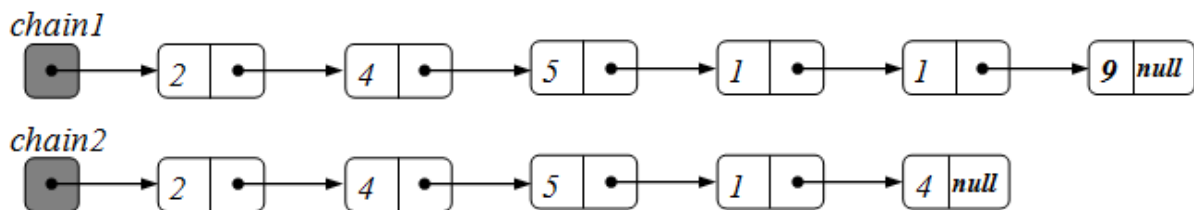
```
public static bool Sod3(Node<int> chain1, Node<int> chain2)
{
    Node<int> pos1 = chain1;
    while (pos1 != null)
```

```

{
    bool found = false;
    Node<int> pos2 = chain2;
    while (pos2 != null && !found)
    {
        if (pos1.GetInfo() == pos2.GetInfo())
            found = true;
        pos2 = pos2.GetNext();
    }
    if (!found)
        return false;
    pos1 = pos1.GetNext();
}
return true;
}

```

א. עקוב אחר כל אחת מהפעולות *Sod1*, *Sod2*, *Sod3*, עם שתי שרשרות החוליות של מספרים שלמים *chain1* ו-*chain2* שלפניך:



רשום מה יוחזר בעבור כל אחת מהפעולות. במעקב הראה את המעבר על שרשרות החוליות.

ב. מה סיבוכיות זמן הריצה של כל אחת מהפעולות *Sod1*, *Sod2*, *Sod3*? נמק את תשובתך.

הצעה 2 רחלי צרניחוב

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

במקום $(List<int> list1, List<int> list2)$

יהיה כתוב $(Node<int> list1, Node<int> list2)$

במקום $Node<int> node1 = list1.GetFirst();$

יהיה כתוב $Node<int> node1 = list1.~~GetFirst()~~;$

```

public static bool Sod1(Node<int> list1, Node<int> list2)
{
    Node<int> node1 = list1;
    Node<int> node2 = list2;

```

```

for (int i = 1; i <= 4; i++)
{
    if ((node1 == null) || (node2 == null))
        return
        false;
    if ((i == 1) || (i == 4))
        if (node1.GetInfo() != node2.GetInfo())
            return
            false;
    node1 = node1.GetNext();
}
return true;
}
public static bool Sod2(Node<int> list1, Node<int> list2)
{
    Node<int> node1 = list1;
    Node<int> node2 = list2;
    while ((node1 != null) && (node2 != null))
    {
        if (node1.GetInfo() != node2.GetInfo())
            return
            false;
        node1 = node1.GetNext();
        node2 = node2.GetNext();
    }
    return true;
}
public static bool Sod3(Node<int> list1, Node<int> list2)
{
    Node<int> node1 = list1;
    while (node1 != null)
    {
        bool found = false;
        Node<int> node2 = list2;
        while ((node2 != null) && (!found))
        {
            if (node1.GetInfo() == node2.GetInfo())
                found = true;
            node2 = node2.GetNext();
        }
        if (!found)
            return false;
        node1 = node1.GetNext();
    }
    return true;
}
}

```


הצעה 3 משה ניסים

מבחן 2013 שאלה 1 גרסת C# פונקציית Sod1

```
public static bool Sod1(List<int> list1, List<int> list2)
{
    Node<int> node1 = list1.GetFirst();
    Node<int> node2 = list2.GetFirst();
    for (int i = 1; i <= 4; i++)
    {
        if ((node1 == null) || (node2 == null))
            return false;
        if ((i == 1) || (i == 4))
            if (node1.GetInfo() != node2.GetInfo())
                return false;
        node1 = node1.GetNext();
    }
    return true;
}
```

השינוי עפ"י סעיף 6 בתוכנית הלימודים החדשה :

"רשימה- ככלל, מבנה הנתונים "רשימה" ייוצג וימומש בתוכנית הלימודים ע"י מחלקת Node ותו לא."

ומאחר שהורדנו את ה- List, כאשר נפנה למשתנים list1, list2 אין לנו צורך יותר ב- GetFirst() משום שאוטומטית ניגש לראשון.

```
public static bool Sod1(Node<int> list1, Node<int> list2)
{
    Node<int> node1 = list1;
    Node<int> node2 = list2;
    for (int i = 1; i <= 4; i++)
    {
        if ((node1 == null) || (node2 == null))
            return false;
        if ((i == 1) || (i == 4))
            if (node1.GetInfo() != node2.GetInfo())
                return false;
        node1 = node1.GetNext();
    }
    return true;
}
```

שאלה 3 סעיף ב, בגרות קיץ תשע"ג 2013

ולרי פקר

הניסוח המקורי:

(1) כתוב ב-Java או ב-C# פעולה שתקבל מספר שלם num, גדול מ-1, ורשימה lst המכילה מספרים שלמים גדולים מ-0, שכולם קטנים מ-num. הפעולה תחזיר רשימה חדשה שאיבריה הם כל המספרים השלמים הגדולים מ-0 וקטנים מ-num, שאינם מופיעים ברשימה lst.

(2) מהי סיבוכיות זמן הריצה של הפעולה שכתבת? נמק את תשובתך.

הצעה לניסוח חדש שיתאים לתכנית "מבני נתונים":

- (1) כתוב ב-Java או ב-C# פעולה שתקבל מספר שלם num, גדול מ-1, ושרשרת של חוליות chain המכילה מספרים שלמים גדולים מ-0, שכולם קטנים מ-num. הפעולה תחזיר הפניה לשרשרת חוליות חדשה שאיבריה הם כל המספרים השלמים הגדולים מ-0 וקטנים מ-num, שאינם מופיעים בשרשרת החוליות chain.
- (2) מהי סיבוכיות זמן הריצה של הפעולה שכתבת? נמק את תשובתך.