

# קורס חינוך השש שנתי

## פרק המדע, נס ציונה



### החוליה הכלכלית $\text{Node} < \text{T} \rangle$

חתימת הפעולה	תיאור הפעולה
	<b>בנייה:</b>
<code>Node (T x)</code>	פעולה הבונה חוליה שבערך <code>value</code> שלה יהיה <code>x</code> וב- <code>next</code> שלה יהיה הערך <code>null</code>
<code>Node (T x , Node&lt;T&gt; next)</code>	פעולה הבונה חוליה שבערך <code>value</code> שלה יהיה <code>x</code> וב- <code>next</code> שלה יהיה הערך <code>next</code> (ערך <code>next</code> המועבר כפרמטר יכול להיות גם <code>null</code> )
	<b>שאלות:</b>
<code>T GetValue ()</code>	אם <code>T</code> מחלקת עוטפת לטיפוס בסיסי (Integer, Double, Character) ומשם הפניה לעצם, תוחזר הפניה לעצם זה יוחזר ערך החוליה, ואם הפניה לעצם, תוחזר הפניה לעצם זה
<code>Node&lt;T&gt; GetNext ()</code>	מוחזרת הפניה לחוליה הבאה
<code>bool HasNext ()</code>	פעולה המחזיר את אמת <code>next</code> מפנה לחוליה נוספת (כלומר אין <code>null</code> ) ושקר אחרת
<code>string ToString ()</code>	פעולה המחזיר מחרוזת המתארת את מצב העצם <b>(*) סיבוכיות:</b> אם <code>T</code> עצם מטיפוס פשוט $\leftarrow O(1)$ ואם <code>T</code> מייצג אוסף $\leftarrow O( T )$ (אורך תואוסף כפונקציה של <code>T</code> )
	<b>פוקודות:</b>
<code>void SetValue (T x)</code>	הפעולה משנה (מעדכנת) את ערך ה- <code>value</code> של החוליה ל- <code>x</code>
<code>void SetNext (Node&lt;T&gt; next)</code>	הפעולה משנה את ערכו של <code>next</code> להיות חדש (ערך <code>next</code> המתקבל כפרמטר יכול להיות גם <code>null</code> )

```
public class Node<T>
{
    private T value;
    private Node<T> next;

    2 references
    public Node(T value) { this.value = value; this.next = null; }
    0 references
    public Node(T value, Node<T> next) { this.value = value; this.next = next; }
    1 reference
    public T GetValue() { return this.value; }
    0 references
    public void SetValue(T value) { this.value = value; }
    8 references
    public Node<T> GetNext() { return this.next; }
    5 references
    public void SetNext(Node<T> next) { this.next = next; }
    2 references
    public bool HasNext() { return (this.next != null); }
    0 references
    public override string ToString() { return this.value.ToString(); }
}
```

תיאור	שם הפעולה
הפעולה משכפלת את הרשימה ומחזירה מבצעי רישימה חדשה.	CopyList (Node<string> head)
הפעולה מחזירה מבצעי לרשימה חדשה בסדר הפוך (החוליה האחורונה היא הראשונה)	Reverse (Node<double> head)
הפעולה מ Chapman את האיבר הראשון עם הערך שהתקבל ומחזירה את המיקום שלו ברישימה.	Index (Node<int> head, int value)
הפעולה מ Chapman את האיבר הראשון עם הערך שהתקבל ומחזירה את האיבר (מצביע לאיבר).	Find (Node<string> head, string value)
הפעולה מחזירה תת רישימה של הרשימה שהתקבלה מן האיבר start ועד לאיבר end. (אין צורך לבדוק תקינות האינדקסים).	Slice (Node<char> head, int start, int end)
<p>א. הפעולה מקבלת שתי רשימות ובודקת אם הערכים שלהם זהים <b>ומסודרים באותו סדר</b>. אם כן, מחזירה אמת. אחרת שקר.</p> <p>ב. הפעולה מקבלת שתי רשימות ובודקת אם הערכים שלהם זהים <b>ולא דוקא מסודרים באותו סדר</b>. אם כן, מחזירה אמת. אחרת שקר.</p>	Equal1(Node<int> L1, Node<int> L2) Equal1(Node<int> L1, Node<int> L2)
הפעולה מקבלת שתי רשימות ומחזירה רישימה מאוחצת L2 אחרי L1. אין צורך להעתיק את האיברים (שימוש באיברים הקיימים).	Union(Node<double> L1, Node<double> L2)
הפעולה מקבלת שתי רשימות ומחזירה רישימה שבה האיברים בעלי הערך הזהה.	Intersect(Node<string> L1, Node<string> L2)
הפעולה מוחקת מן הרשימה את האיברים הכפולים ומשאירה רק איבר אחד מכל ערך.	Unique(Node<int> head)
הפעולה מחזירה את האיבר עם הערך הגדול ביותר. אתם יכולים להניח כי ניתן להשוות בין הערכים באמצעות < או >.	Max(Node<int> head)
הפעולה מקבלת שתי רשימות ממוינות (אתם יכולים להניח כי ניתן להשוות בין הערכים באמצעות < או >). הפעולה מחזירה רישימה ממוגנת וממוינת (אין להעתיק איברים).	UnionSort(Node<double> L1, Node<double> L2)

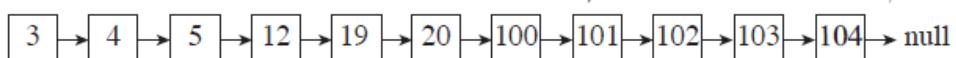
<p>הפעולה מקבלת רשימה של int שהערכים בין 1 ל- 100. הפעולה תחזיר את המספר השכיח ביותר (שמופיע הכי הרבה פעמים). ניתן להשתמש במבנה נתונים נוסף.</p>	<p>commonValue (Node&lt;int&gt; head)</p>
<p>הפעולה מקבלת רשימה ומחזירה אמת היא ערבי האיברים עולים ממש או יורדים ממש. אתם יכולים להניח כי ניתן לבצע השוואה בין האיברים באמצעות &lt; או &gt;).</p>	<p>Consistent(Node&lt;int&gt; head)</p>
<p>הפעולה מקבלת רשימה ומחזירה true אם היא פולינדרום. רמז : ניתן להשתמש בשתי פעולה קודומות שכתבתם והתשובה מסכמת בשלוש שורות.</p>	<p>isPalindrome(Node&lt;char&gt; head)</p>
<p>הפעולה מקבלת שתי רשימות ומחזירה אמת אם רשימה L2 מוכלת ב L1 . <math>L2 \subseteq L1</math></p>	<p>Contains(Node&lt;int&gt; L1, Node&lt;int&gt; L2)</p>
<p>הפעולה מקבלת שתי רשימות ומאחדת אותן לרשימה אחת בצורת ריצ'רץ (איבר מ L1 , איבר מ L2 וחזר חלילה).</p>	<p>Zipper (Node&lt;string&gt; L1, Node&lt;string&gt; L2)</p>
<p>פעולה מקבלת רשימה של מספרים ממשיים, יוצרת רשימה של המספרים החובבים מתוך הרשימה, מדפסה (באמצעות PrintList() ) את הרשימה החדשה ולאחריה את הרשימה המקורי, ומחזירה את הרשימה החדשה.</p>	<p>Positive(Node&lt;double&gt;)</p>
<p>הפעולה מקבלת שרשרת חוליות של מספרים שלמים ומחזירה שרשרת חדשה כך : עבור כל תת-רשימה של מספרים עולים ב L , שבה כל מספר גדול ממש מקודמו, יופיע ברשימה החדשה איבר אחד עם סכום תת הסדרה. כל תת סדרה מסתויימת כאשר אחר האיבר האחרון מופיע איבר שווה או קטן לו.</p>	<p>SumOfsubList(Node&lt;int&gt; L)</p>
<p>רשימה L תיקרא משולשת אם היא מקיימת את שלושת התנאים הבאים : (א) הרשימה אינה ריקה ; (ב) מספר האיברים בה מחלק ב-3 ; (ג) האיברים בשליש הראשון של הרשימה מכילים את אותם ערכים שמכילים האיברים בשליש השני ובשליש השלישי ובאותו הסדר. כתוב פעולה מקבלת רשימה ומחזירה אמת אם היא משולשת אחרת שקר.</p>	<p>isTripleList (Node&lt;int&gt; L)</p>
<p>הפעולה מקבלת רשימה ממוקנת של int ומוסיפה איבר עם הערך החדש במקום המותאים. אם הערך קיים ניתן להוסיפו לפני או אחרי החוליה השווה לו.</p>	<p>addSort(Node&lt;int&gt; L, int n)</p>

<p>הפעולה מקבלת רשימה של מספרים שלמים ומוחזירה רשימה חדשה ממוינת (הרשימה המקורי לא משתנה). ניתן וכדי לעשוט שימוש בפעולות קודמות.</p>	<p>sortList(Node&lt;int&gt; L)</p>
--	------------------------------------

## שאלה מבחן 2010

2. L היא רשימה המכילה מספרים שלמים שונים זה מזה וממוינים בסדר עולה.  
**רשימת הטוווחים של L** היא רשימה חדשה שנבנית באופן הזה: בעבר בלרצף של מספרים עוקבים ב- L יהיה **ברשימה הטוווחים 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 ברשימה הטוווחים על ידי איבר שני המספרים בו שווים.

לדוגמה, בעבר הרשימה L שלפניך:



**רשימת הטוווחים של L** תהיה:



לפניך תיאור חלקו של המחלקה **RangeNode**, המייצגת איבר **ברשימה הטוווחים**.

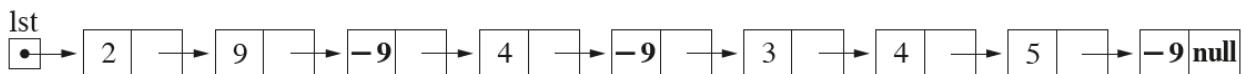
RangeNode
private int from; // המספר הקטן ביותר ברצף
private int to; // המספר הגדול ביותר ברצף
public RangeNode(int from, int to)

מבקש ב- Java או ב- C# פעולה חיצונית שתקבל רשימה לא ריקה, המכילה מספרים שלמים שונים זה מזה וממוינים בסדר עולה, ותחזיר את **רשימת הטוווחים** שלה.

public static Node<RangeNode> CreateRangeList(Node<int> sourceList)

## שאלה מבגרות 2020

- 4. "שרשרת מספרים שלמים חיוביים" היא שרשרת חוליות שכל חוליה בה מכילה מספר שלם גדול מד 0 .
  - "שרשרת ספרות" היא שרשרת חוליות שכל חוליה בה מכילה ספרה בין 0 ל- 9 (כולל) או את המספר (9) – .
  - כל רצף ספרות בשרשראת מייצג מספר: הספרה הראשונה מייצגת את האחדות, הספרה השנייה את העשרות וכן הלאה. לאחר כל רצף של ספרות מופיע המספר (9) – , והוא מסמן סוף של מספר בשרשראת.
- לפניך דוגמה ל"שרשרת ספרות" המייצגת את המספרים: 92 , 4 , 543 .



כתב פעולה חיצונית בשפת Java בשם buildDigit # בשמם buildDigit , מקבלת הפניה lst שאינה null ל"שרשרת מספרים שלמים חיוביים". הפעולה תחזיר "שרשרת ספרות" המייצגת את המספרים שב"שרשרת מספרים שלמים חיוביים" לפי הסדר.

## שאלה ברמת בוגרים

משמעות הפעולה הבאה ונתחו את ייעולותה :

```
public static int maxSortedSubList(List<Integer> lst)
```

הפעולה מקבלת רשימה של מספרים שלמים ומחזירה את אורך התת-רשימה הרציפה הארוכה ביותר המסדרת בסדר עולה ממש.

**דוגמאות :**

- עבור הרשימה : maxSortedSubList(lst) ,lst = 3, 1, 2, 3, 2, -3, -1, 2, 4, 7 .  
התת-רשימה העולה הארוכה ביותר, היא התת-רשימה המתחילה ב : -3 – ומסתיימת ב : 7 , ויש בה חמישה איברים.
- עבור הרשימה : maxSortedSubList(lst) ,lst = 10, 7, 5, 3, -2, -30 .  
שלא קיימת תת-רשימה עולה שאורכה 2 לפחות.
- עבור הרשימה : maxSortedSubList(lst) ,lst = -4, 0, 1, 6, 12, 23, 90 .  
התת-רשימה העולה הארוכה ביותר היא כל הרשימה, ויש בה 7 איברים.

## תור Queue

```
public class Queue <T>
{
    private Node<T> first;
    private Node<T> last;
    0 references
    public Queue() ...
    0 references
    public bool isEmpty () ...
    0 references
    public T Head() ...
    0 references
    public void Insert (T value) // מוסיף איבר לסוף התור ...
    0 references
    public T Remove () // מוציא ומחזיר את האיבר הראשון בתור ...
    0 references
    public override string ToString() ...
}
```

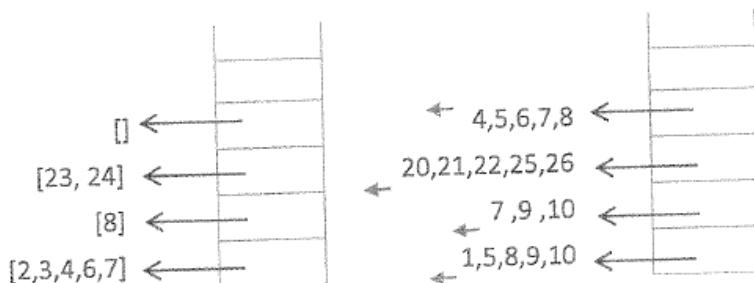
מהות	הפעולה
פעולה גנריית חיצונית המקבלת מערך של $<T>$ ויצרת ממנו תור.	CreateQueue $<T>$
שרשור – פעולה גנריית המקבלת שני תורים $q_1, q_2$ ומשרשרת אותם לתור אחד $q$ (מחברת את $q_1$ עם $q_2$ ).	Concatenate $<T>$
פעולה גנריית המקבלת תור וערך ובודקת אם הערך קיים בתור. התור אינו משתנה. שימוש בתור עוזר	isExist1 $<T>$
פעולה גנריית המקבלת תור של איברים שונים זה מזה וערך, ובודקת אם הערך קיים בתור. התור אינו משתנה ואין להשתמש במבנה נתונים נוסף.	isExists2 $<T>$
פעולה המקבלת תור של מספרים חיוביים וערך, ובודקת אם הערך קיים בתור. התור אינו משתנה ואין להשתמש במבנה נתונים נוסף.	isExist3
פעולה גנריית חיצונית המקבלת תור ומחזירה העתק של תור. המחסנית המקורית אינה משתנה.	Copy
פעולה גנריית המקבלת תור והופכת את סדר אבוריו.	Reverse $<T>$
פעולה גנריית המקבלת תור של מספרים שלמים ומוציאה ומחזירה את המספר המינימלי. התור לא ישנה למעט הוצאת המספר המינימלי. אם יש מספר מספרים מינימליים זהים הפעולה תוצאה רק אחד מהם.	PopMin
כתבו פעולה המקבלת תור צו זר ומחזירה תור חדש המומן מהמספר הקטן לגודל (המספר הקטן בראש המחסנית). בסיום הפעולה התור המקורי לא משתנה.	Sort
כתבו פעולה המקבלת תור של מספרים שלמים ומספר שלם צ. הפעולה תחזיר true אם מספר החזרות של כל אחד מהמספרים בטוחה מ-1 הוא כערך המספר עצמו : המספר 1 מופיע פעם אחת, המספר 2 מופיע פעמיים וכך הלאה. אין חשיבות לסדר המספרים. אין להשתמש באלגוריתם למיון. התור לא ישנה. (שאלה 5 בספר).	Sequence
כתבו פעולה חיצונית המקבלת תור המכיל מספרים טבעיים שונים זה מזה. על הפעולה להחזיר מחסנית ממוניה בסדר גודול ביותר בראש המחסנית) הבניה מכל אברי התור, תוך התחשבות במוגבלות הבאות : (1) בשום שלב אין לשלו איברים מהחסנית ; (2) אין להשתמש במבנה נתונים נוספים למעט התור שהתקבל.	SortedStack
כתבו פעולה חיצונית המקבלת שני תורים $q_1, q_2$ של מספרים ממשיים double וממוינים מהקטן לגודל. הפעולה תמזג אותם לתוך $q$ כתור יחיד ממויין.	Merge

כתבו פוליה המתקבלת תור של תלמידים, ומחזירה מחסנית של התלמידים עם הציון הכי גבוה (יכולים להיות מספר תלמידים שקיבלו את הציון הכי גבוה). הערך הוא אובייקט Student הכלול string name int grade . הניחו שיש פעולות get, set במחלקה ובנאים כפי שתבחרו. סיבוכיות הזמן חייבת להיות  $O(n)$ .

bestStudents

### תור של תורים ותור של שרשרת חוליות

נתון תור QQ שבו כל איבר הוא תור של מספרים שלמים. כל תור ממוקם בסדר עולה כך שהאיבר בראש התור הוא הקטן ביותר.  
לדוגמה :



- א. כתבו פוליה חיצונית המתקבלת תור של תורים QQ ומחזירה תור שכל איבר בו הוא שרשרת חוליות. QL הוא תור משלים לתור QQ כך שבכל שרשרת הממוקמת בתור QL נמצאים כל המספרים החסרים בתור הממוקם באותו מקום בתור QQ, כדי שהיה רצוי של מספרים עוקבים מהקטן ועד הגדול. על המספרים בשרשראות להיות ממוקנים מהקטן לגדול כך שהמספר בתחילת השרשראת הוא הקטן ביותר.
- ב. מה סיבוכיות זמן ריצה של הפוליה שכתבתם ? נמקו

## שאלה מוגרות 2021

.7

**a.** מימוש את הפעולה `Size` המתווארת בטבלה הבאה.

כותרת הפעולה	תיאור הפעולה
<code>public static int size (Queue&lt;Integer&gt; q)</code> – <b>Java</b>	הפעולה מחזירה את מספר האיברים בתור <code>q</code>
<code>public static int Size (Queue&lt;int&gt; q)</code> – <b>C#</b>	בליל לשנות את התור.

**b.** שני תורים, `q1` ו-`q2`, יהיו "תורים זחים" אם מספר האיברים בשני התורים זהה, ובשני התורים מופיעים בד同 אותם ערכים ובאותו הסדר.

דוגמיה לשני תורים זחים:

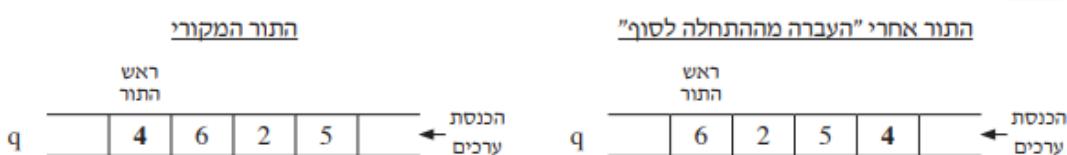


כתב פוליה חיצונית ששם `isIdentical` בשפת Java או `IsIdentical` בשפת C# מקבלת שני תורים מטיפוסם שלם `q1` ו-`q2`, וממחישה `true` אם התורים זחים, אחרת היא מחזירה `false`.

הערה: עם סיום הפעולה, חובב לשומר על מבנה התורים המקורי שהתקבל.

"העברה מההתחלת לסוף" היא העברת מספר מראש התור לסופו.

דוגמיה:



כתב פוליה חיצונית ששם `isSimilar` בשפת Java או `IsSimilar` בשפת C# מקבלת שני תורים מטיפוסם שלם, `q1` ו-`q2`. הפעולה מחזירה `true` אם התורים `q1` ו-`q2` זחים – בין שהם זחים כמו שם ובין שהם יהיו זחים לאחר שנבצע ב-`q1` "העברה מההתחלת לסוף", בעממי. אחרת הפעולה מחזירה `false`.

דוגמיה: הפעולה תחזיר `true` בעבור שני התורים `q1` ו-`q2` שלפנינו:



זאת מכיוון שלאחר שבתור `q1` נבצע בעממי "העברה מההתחלת לסוף", הוא יהיה זהה לתור `q2`, ויראה כך:

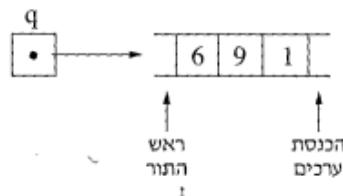


הערות: – חובב להשתמש בפעולת `ShiftLeft` בסעיף A.

– אין צורך לשמנון על מבנה התורים המקורי שהתקבל.  
/המשך בעמוד 18/

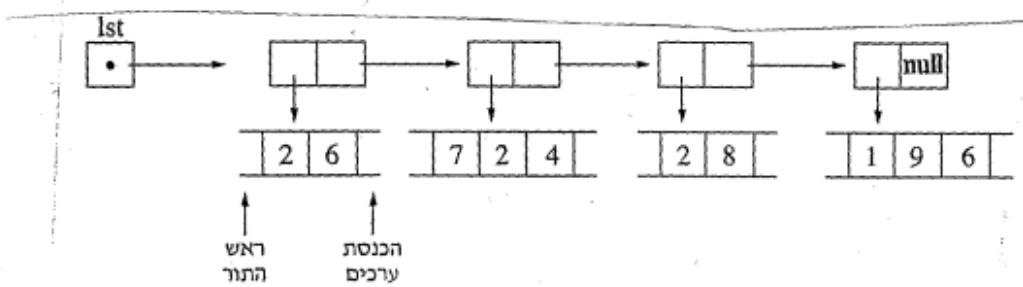
## שאלה מבגרות 2019

- א. "תור מספר" הוא תור של ספורות בין 1 ל-9 (כולל), המיצג מספר שלם – האיבר הראשון (ראש התור) הוא ספרת הначדונם, האיבר השני היא ספורה העשרות וכן הלאה.  
הנוכח שמספר הספורות האפשרי בתור לא גדול ממספר הספורות שיכל למכיל משתנה מטיפוס `int`.  
לדוגמא: התור שלפניך מייצג את המספר 196.



כתב פוליה בשם `toNumber` ב- C# או `ToNumber` ב- Java או `toNumber` ב- C#, מקבלת "תור מספר" – `q`, ומוחזירה את המספר המיוצג בתור.  
הערה: אין חובה לשומר על מבנה התור.

- ב. נתונה שרשרת חוליות ובה כל חוליה מכילה "תור מספר".  
לדוגמא: שרשרת החוליות שלפניך מייצגת את המספרים 691, 82, 427, 62.



כתב פוליה בשם `bigNumber` ב- C# או `bigNumber` ב- Java או `BigNumber` ב- C#, מקבלת הפניה `lst` לשרשורת החוליות, ומוחזירה את המספר הגדל ביותר המיוצג בשרשורת החוליות.  
עבור שרשרת החוליות שתוארה בדוגמה הפעולה מוחזירה את המספר 691.  
חובה להשתמש בפולה שהוגדרה בסעיף א.

## שאלה מבחן 2006

.1. לפניך פועלה:

<b>תור _לפי_ שכיחות (Q)</b>
<p>הפעולה מקבלת תור Q המכיל מספרים שלמים, ומחזירה תור חדש. בעבור כל מספר בתור Q, יהיה בתור החדש שני איברים: האיבר הראשון מכיל את המספר מהתור Q, והאיבר השני מכיל את מספר הפעמים שהוא מופיע בתור Q. בעבור מספר המופיע יותר מפעם אחת בתור Q, יהיה זוג אחד בלבד בתור החדש.</p> <p>הנחה: התור Q מאותחל.</p> <p>הערה: אין לשנות את התור Q.</p>

לדוגמה:

נתון התור Q (משמאל לימין):

1	4	4	1	5	-9	1	-9	-9
---	---	---	---	---	----	---	----	----

התור שיוחזר לאחר זימונן הפעולה **תור \_לפי\_ שכיחות (Q)** יהיה (משמאל לימין):

1	3	4	2	5	1	-9	3
---	---	---	---	---	---	----	---

a. כתוב אלגוריתם, שימש את הפעולה **תור \_לפי\_ שכיחות (Q)**.

אפשר להשתמש בפעולות המשק תור ובפעולה **העתק\_טור (Q)** שלפניך,

בלי למשמש אותן.

<b>העתק_טור (Q)</b>
<p>הפעולה מקבלת תור Q וממחזירה תור חדש זהה לו.</p> <p>הנחה: Q מאותחל.</p> <p>סיבוכיות זמן הריצה: <math>O(n)</math>, n הוא מספר האיברים בתור Q.</p>

b. מהי סיבוכיות זמן הריצה של האלגוריתם שכתבת בסעיף a? נמק את תשובהך.

הנחה שסיבוכיות זמן הריצה של כל אחת מפעולות המשק תור היא  $O(1)$ .

/המשך בעמוד 3/

+

## שאלה מבגרות 2009 – רקורסיה

לפניך 2 פעולות חיצונית הכתובות ב- C# .

הפעולה מקבלת תור לא ריק, המכיל מספרים שלמים. \*/  
/\* הפעולה מחזירה...

```
public static int Sod1 (Queue<int> q)
{
    int i = q.Remove ();
    int result = i;

    if (!q.IsEmpty ())
    {
        int j = Sod1 (q);
        if (result > j)
            result = j;
    }
    q.Insert (i);
    return result;
}
```

הפעולה מקבלת מספר שלם גדול מ- 0 או שווה לו \*/  
/\* הפעולה מחזירה...

```
public static int Sod2 (int i)
{
    if (i == 0)
        return 0;
    int a = i % 10;
    int b = Sod2 (i / 10);
    if (a > b)
        return a;
    return b;
}
```

## עצים בינאריים

```

public class BinNode <T>
{
    private BinNode<T> left;
    private T value;
    private BinNode<T> right;

    0 references
    public BinNode (T value) ...
    0 references
    public BinNode (BinNode<T> left, T value, BinNode<T> right) ...

    0 references
    public T GetValue () { return this.value; }
    0 references
    public BinNode<T> GetLeft () { return this.left; }
    0 references
    public BinNode<T> GetRight () { return this.right; }
    0 references
    public bool HasLeft () { return this.left != null; }
    0 references
    public bool hasRight () { return this.right != null; }
    0 references
    public void SetValue (T value) { this.value = value; }
    0 references
    public void SetLeft (BinNode<T> left) { this.left = left; }
    0 references
    public void SetRight (BinNode<T> right) { this.right= right; }
    0 references
    public override string ToString() {return this.value.ToString();}

}

public static void graphicTree ()
{
    BinNode<int> t = BinTreeUtils.BuildRandomTree(50, 1, 3);
    TreeCanvas.AddTree(t);
    TreeCanvas.TreeDrawPostOrder();
}

```

### מחלקה גרפית לייצוג עצ' בינירי

הקובץ *Unit.dll* מכיל את מחלקת השירות *TreeCanvas* במרחב השמות *.Unit4.CanvasLib*. מחלקת השירות מאפשרת הצגה גרפית של עצים בינריים המבוססים על המחלק הגנרי *BinNode<T>*, וכן מוגנות השירותים הבאים לפי העץ שהתקבל: סריקה תחילה, סריקה תוכית, סריקה סופית וסריקה לפי רמות.

<code>static void AddTree&lt;T&gt;(BinNode&lt;T&gt; tree)</code>	הפעולה מעבירה את העץ הבינירי המתתקבל כפרמטר לחalon גרפִי וגם מציגה אותו.
<code>static void SetAnimastion(AnimationSpeed speed)</code>	הפעולה קובעת מהירות של אנימציה בסריקה של העץ הבינירי שהועבר לחalon גרפִי על-ידי הפעולה <i>AddTree</i> .
<code>static void PrintOutAllElements(bool printOut)</code>	הפעולה קובעת האם להציג תוצאות הסריקה בחלון <i>Console</i> , של העץ הבינירי שהועבר לחalon גרפִי על-ידי הפעולה <i>AddTree</i> .
<code>static void TreeDrawInOrder()</code>	פעולה מתארת את סריקת העץ הבינירי שהועבר לחalon גרפִי בסדר תaci.
<code>static void TreeDrawPostOrder()</code>	פעולה מתארת את סריקת העץ הבינירי שהועבר לחalon גרפִי בסדר סופי.
<code>static void TreeDrawPostOrder()</code>	פעולה מתארת את סריקת העץ הבינירי שהועבר לחalon גרפִי בסדר תחيلي.
<code>static void TreeDrawLevelOrder()</code>	פעולה מתארת את סריקת העץ הבינירי שהועבר לחalon גרפִי לפי רמות.

```
***** Tree Print*****
public static void BinTreeMethods()
{
    string path = GetTreeFilePath("tree_new.txt");
    BinNode<int> tree = BuildBinaryTree<int>(path);
    PrintBinaryTree(tree);
    PrintBinaryTreeColored(tree);
}
```

```

tree_new.txt ✎ ×
1          10
2          Left:3
3          Left:4
4          Left: 0
5          Right: 2
6          Right:6
7          Left:1
8          Right:27
9          Right: 4
10         Left:2
11         Right:3
12

```

## סדר הסריקה לעומק DFS

```
public static void DFS <T> (BinNode<T> node)
{
    if (node == null)
        return;
    Console.Write(node + ", ");
    DFS(node.GetLeft());
    DFS(node.GetRight());
}
```

```
public static void DFS <T> (BinNode<T> node)
{
    if (node == null)
        return;
    // preOrder - תחילה
    DFS(node.GetLeft());
    // inOrder - תובית
    DFS(node.GetRight());
    // postOrder - סוף
}
```

הפעולה	תיאור
	התקן את הממשקים הנדרשים. צור קובץ עם עץ <root> בעל שלוש רמות לבחירתך. הדפס את העץ באמצעות הספרייה הגרפית.
DFS	פעולה מקבלת עץ (שורש - חוליה בינרית) ומחזירה תור עם סדר הערכיים בסריקה תחילית
DFS_count	הפעולה סופרת את מספר הצמתים בעץ.
DFS_FindMax	הפעולהמחזירה את ערך הצומת הגבוה ביותר בעץ.
DFS_FindMaxLeaf	הפעולהמחזירה את ערך העלה הגבוה ביותר בעץ.
DFS_depth	הפעולהמחזירה את עומק העץ
DFS_sum_of_nodes	הפעולה סוכמת את כל הערכיים של צמחי העץ.
DFS_sum_of_leaves	הפעולה סוכמת את כל הערכיים של עלי העץ.
DFS_IsAllEven	הפעולהמחזירה true אם כל העלים זוגיים
DFS_SortSons	הפעולה מקבלת עץ ומסדרת את כל העלים כך שהעליה הקטן יהיה מצד ימין.

## **בגרות 2022**

.7 שאלה בנושא עץ בינארי

בשאלה זו אפשר להשתמש בפעולה החיצונית eraseFirst / EraseFirst שלהן בלי למסח אותה.

דוגמאות	תיאור הפעולה	כותרת הפעולה
, str = "hello" – בעבור המחרוזת "ello" הפעולה תחזיר את המחרוזת "ello".	הפעולה מוחירה תנ"ם מחרוזת str, ללא התו הראשון.	<b>בשפת Java :</b> public static String eraseFirst (String str)
, str = "temp" – בעבור המחרוזת "emp" הפעולה תחזיר את המחרוזת "emp".	אם המחרוזת – str ריקה לפני זימון הפעולה, תהיה שגיאה.	<b>בשפת C# :</b> public static string EraseFirst (string str)
, str = "m" – בעבור המחרוזת "m" הפעולה תחזיר מחרוזת ריקה "".		
, str = "" – בעבור המחרוזת הריקה "" התייה שגיאה.		

משוו את הפעולה החיצונית שלהן:

**Java** – public static boolean wordFromRoot (BinNode<Character> tree, String str)

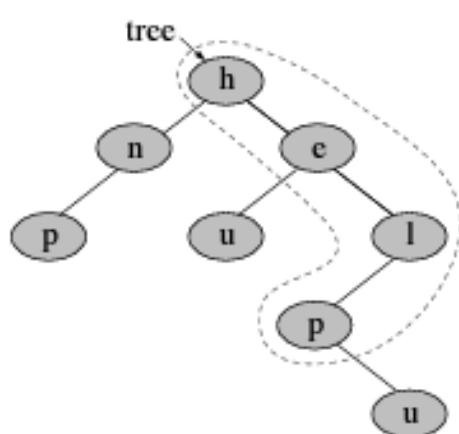
**C#** – public static bool WordFromRoot (BinNode<char> tree, string str)

הפעולה מקבלת מחרוזת – str המכילה לפחות תו אחד, והפניה לעץ בינירי של תווים – tree שאינו null. הפעולה תחזיר true אם קיים מסלול המתחילה בשורש העץ שבו בצת התווים זהה למחרוזת – str. אחרת הפעולה תחזיר false.

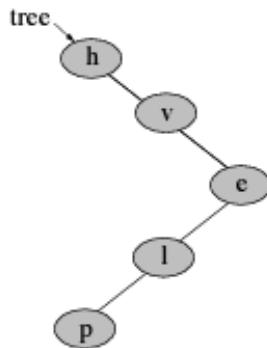
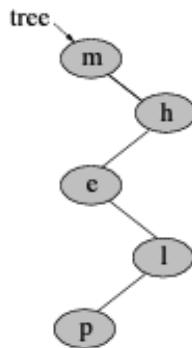
הערה: אות קטנה ואות גדולה אינן זהות זו לזו.

דוגמה:

בעבור העץ הנתון והמחרוזת "help" הפעולה תחזיר true .



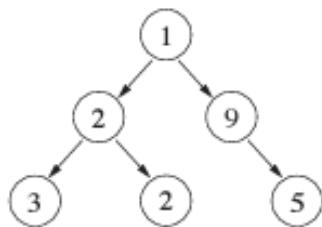
(שים לב: המשך השאלה בעמוד הבא.)

דוגמה:בעבור העץ הנתון והמחזרות "help" הפעולה תחזיר `false`.הסבר: לא קיים בעץ רצף תווים זהה למחרוזת "help".דוגמה:בעבור העץ הנתון והמחזרות "help" הפעולה תחזיר `false`.הסבר: אף על פי שקיים בעץ רצף תווים זהה למחרוזת "help", הפעולה תחזיר `false`, כי הרצף אינו מתחילה בשורש העץ.

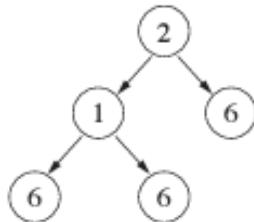
## **בגרות 2020**

6. נגיד: "עַצְמָסוּרִים" הוא עץ בינארי מטיפוס שלם, שכל צומת בו מכיל ספרה בין 1 ל- 9 (כולל), וכל מסלול בעץ מן השורש לעלה מייצג מספר: **העליה מייצגת ספרת האחדות**, הרומה שמעליו את ספרת העשרות וכן הלאה עד השורש של העץ.

**דוגמה:** בעץ המספרים שלפניך מיוצגים המספרים: 195 , 122 , 123 (במסלולים בעץ משמאלי לימין).



**דוגמה נוספת:** בעץ המספרים שלפניך מיוצגים המספרים: 26 , 216 , 216 (במסלולים בעץ משמאלי לימין).

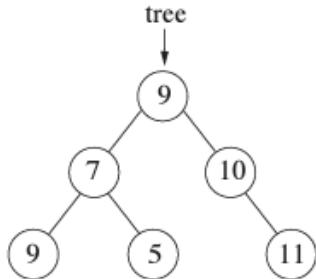


כתב פוליה חיצונית `printAll` בשפת C# או `PrintAll` בשפת Java . הפעולה מקבל עץ מספרים `tree` מטיפוס שלם ותדפיס את כל המספרים שהמסלולים בעץ מייצגים.  
אם `tree` הוא `null` הפעולה לא תדפיס דבר.  
**הערה:** אין חשיבות לסדר שבו המספרים מודפסים.

## בגרות 2020 – מועד מיוחד

6. עץ בינרי מטיפוס שלם של מספרים שאינם שליליים הוא **"עץ שאריות שווני"** במקרה זה:  
 כמות האיברים במספריים מתחלקיים ב- 3 עם שארית 1 שווה לכמות האיברים במספריים מתחלקיים ב- 3  
 עם שארית 2, ושווה לכמות האיברים במספריים מתחלקיים ב- 3 ללא שארית.

דוגמה של **"עץ שאריות שווני"**:

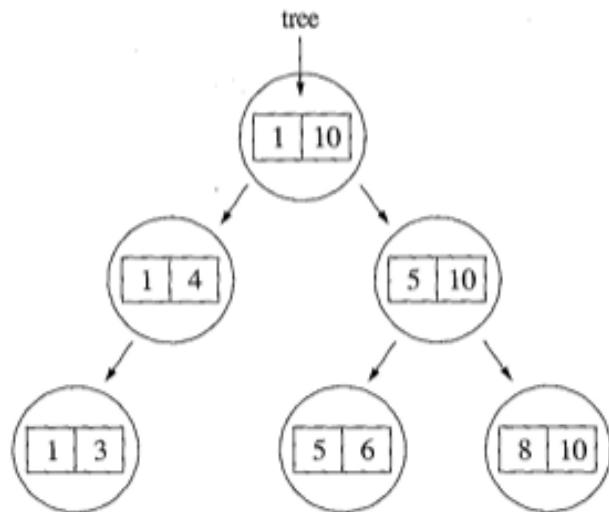


עץ בינרי זה הוא **"עץ שאריות שווני"** מכיוון שהוא שווה ב- 3 במספריים  
 שמתחלקיים ב- 3 עם שארית 1 (7, 10) ומספריים שמתחלקיים ב- 3 עם שארית 2 (9, 5).  
 שני מספריים שמתחלקיים ב- 3 עם שארית 0 (9, 9) ללא שארית.

כתב פעולה חיצונית בוליאנית בשפת Java בשם `treeEqual` או בשפת C# בשם `TreeEqual` מקבלת עץ בינרי  
 מטיפוס שלם, לא ריק, של מספרים שאינם שליליים ובודקת אם הוא **"עץ שאריות שווני"**.  
 אם כן – תחזיר הפעולה `true`, אחרת היא תחזיר `false`.

## בגרות 2019

דוגמיה לעץ טווחים מסודר:



כתב פעלת חיצונית בוליאנית בשם order ב- C# או Order ב- Java, המתקבל עץ טווחים או עץ ריק ומחזירה true אם העץ הוא עץ טווחים מסודר, אחרת – הפעלה מחזירה false.

6. נתונה המחלקה **Range** שיש לה שתי תכונות:

low – מספר מטיפוס שלם.

high – מספר מטיפוס שלם.

high גדול מ- low .

הנח שלכל תכונה הוגדרו ב- Java הפעולות get ו- set וב- C# הפעולות Get ו- Set .

עץ טווחים הוא עץ שאיבריו הם מטיפוס **Range** .

עץ טווחים מסודר הוא עץ ריק או עץ טווחים שבו עבור כל צומת מתקיימים התנאים הבאים:

- אם יש בן שמאל, אז ה- low של הצומת שווה ל- low של הבן השמאלי, וה- high של הצומת גדול או שווה ל- high של הבן השמאלי.
- אם יש בן ימני, אז ה- high של הצומת שווה ל- high של הבן הימני, וה- low של הצומת קטן או שווה ל- low של הבן הימני.
- אם יש שני בניים, אז ה- high של הבן השמאלי קטן מה- low של הבן הימני.

## סרייה לעומק ללא רקורסיה

```
public static void DFS_Stack <T> (BinNode<T> node)
{
    Stack<BinNode<T>> stack = new Stack<BinNode<T>>();
    stack.Push(node);
    while (!stack.IsEmpty())
    {
        node = stack.Pop();
        Console.Write(node + ", ");
        if (node.HasRight()) { stack.Push(node.GetRight()); }
        if (node.HasLeft()) { stack.Push(node.GetLeft()); }
    }
    Console.WriteLine();
}
```

## BFS - סרייה לרוחב ללא רקורסיה

```
public static void BFS<T>(BinNode<T> node)
{
    Queue<BinNode<T>> queue = new Queue<BinNode<T>>();
    queue.Insert(node);
    while (!queue.IsEmpty())
    {
        node = queue.Remove();
        Console.Write(node + ", ");
        if (node.HasLeft()) { queue.Insert(node.GetLeft()); }
        if (node.HasRight()) { queue.Insert(node.GetRight()); }
    }
    Console.WriteLine();
}
```