



קרית חינוך "פארק המדע"
בית לערכמים, למצינות וחדשנות



חוליה - שרשota חוליות רשיימה מקוישרת

כיתה יא

גָלְעֵד מִרְקָמֶן



קריית חינוך "פארק המדע"

בית לערכים, למצוינות וחדשנות

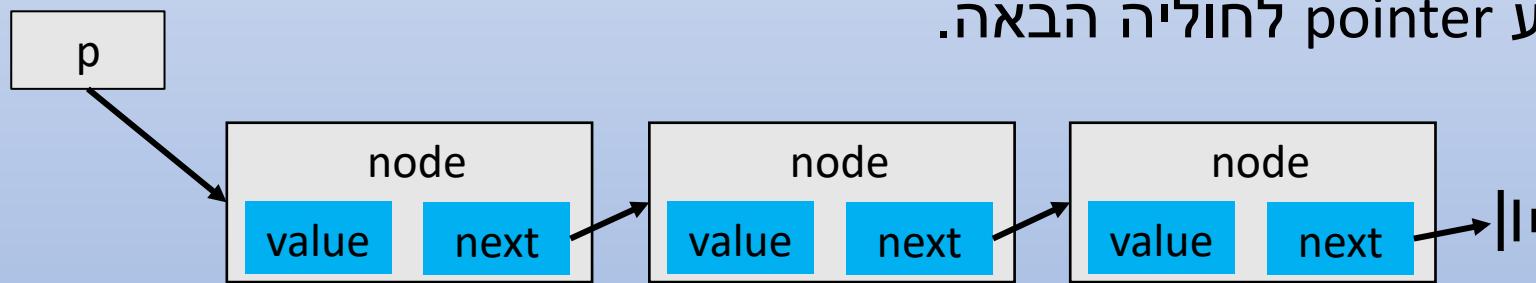
הצורך במבנה נתוניים דינמי

- מערך הינו מבנה נתונים המאפשר להציג מספר רב של נתונים מסוגים שונים. אנו יכולים להגיד מערך של אובייקטים ולשמור בתוכו כל סוג של אובייקט.
- החסרונו במערך הוא שבעת היצירה (האיתחול) המחשב מקצת גודל קבוע בזיכרון שלא ניתן לשנות אותו. ביצירת המערך אנו קובעים את מספר האיברים שייהיו במערך ולא ניתן להגדיל את מספר האיברים במערך תוך כדי ריצת התוכנית.
- בשל כך נוצרו מבני נתונים דינמיים, כגון: רשימה מקושרת, מחסנית, תור, עץ בינארי ועוד.



החוליה Node

- החוליה היא אבן הבניין של מבני נתונים דינמיים רבים.
- חוליה היא אובייקט המורכב משני פרמטרים:
 - value – מקום בו שומרים את הערכים / הנתונים של החוליה. הנתונים יכולים להיות מספר, מחרוזת או אובייקט.
 - Next - מצביע pointer לחוליה הבאה.



- באמצעות שרשור של החוליות אנו יכולים ליצור מבנה נתונים ללא הגבלה, להוסף, ולהוריד חוליות בהתאם לצורך.



חוליה פשוטה

- נגיד לדוגמה חוליה מקבלת ערכים של int, ובנייה שרשרת של שתי חוליות:

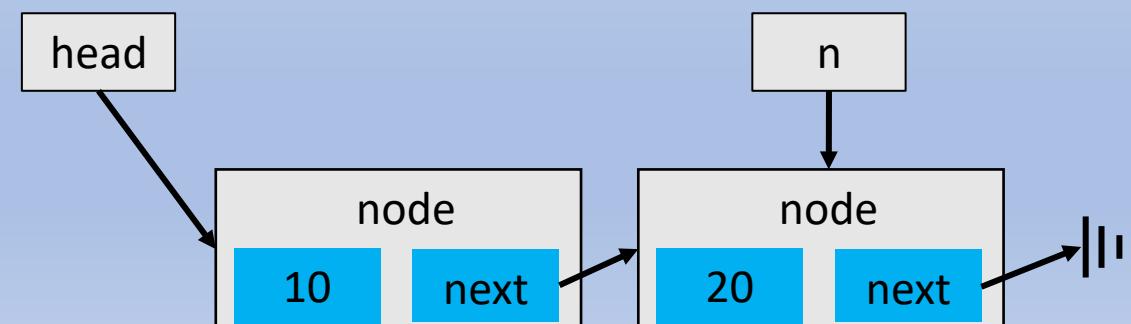
```
class Node
{
    private int value;
    private Node next;

    2 references
    public void SetValue(int value) { this.value = value; }
    2 references
    public int GetValue() { return this.value; }
    1 reference
    public void SetNext(Node node) { this.next = node; }
    1 reference
    public Node GetNext() { return this.next; }
}
```

```
static void Main(string[] args)
{
    Node head = new Node();
    head.SetValue(value: 10);

    Node n = new Node();
    n.SetValue(value: 20);
    head.setNext(node: n);

}
```



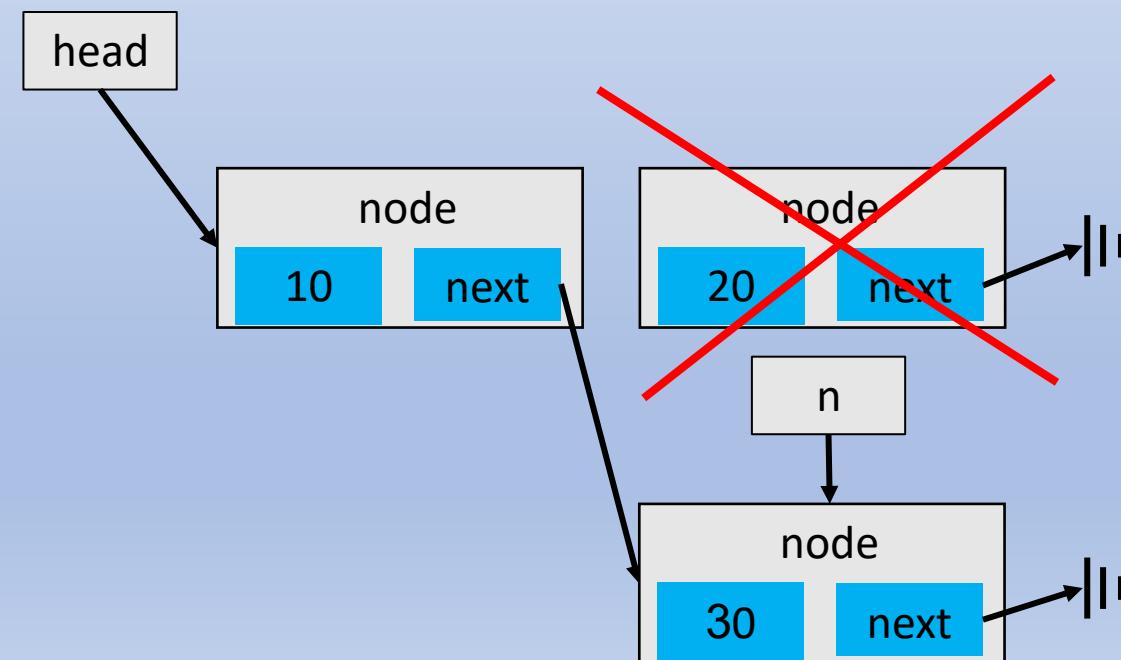
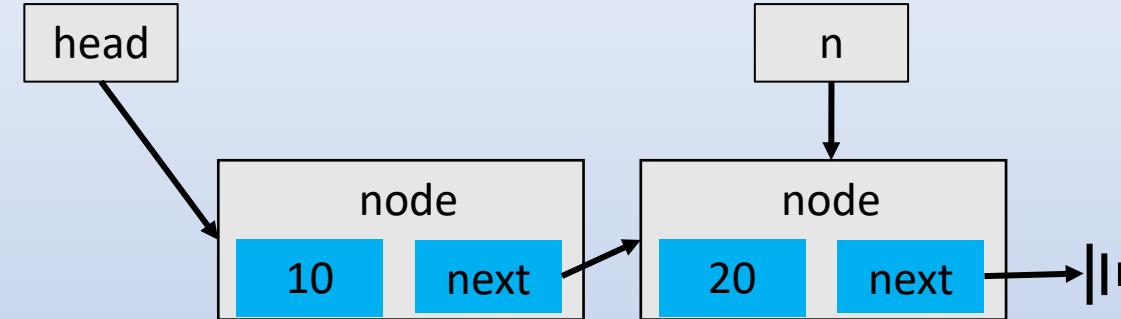


מהי תמונה הזיכרון של הקוד הבא?

```
static void Main(string[] args)
{
    Node head = new Node();
    head.SetValue(10);

    Node n = new Node();
    n.SetValue(20);
    head.SetNext(node: n);

    n = new Node();
    n.SetValue(30);
    head.SetNext(node: n);
}
```



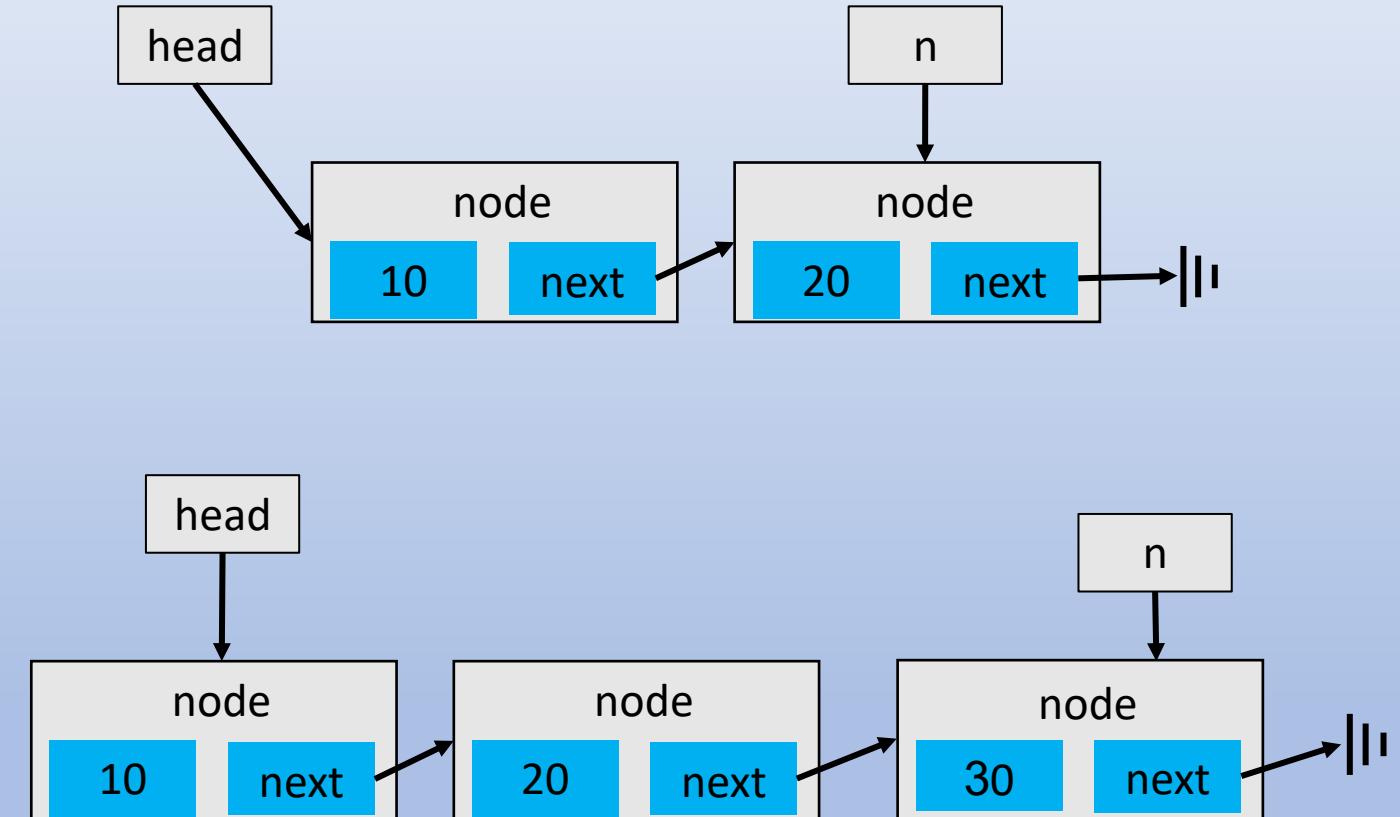


מהי תמונת הזיכרון של הקוד הבא

```
0 references
static void Main(string[] args)
{
    Node head = new Node();
    head.SetValue(10);

    Node n = new Node();
    n.SetValue(20);
    head.SetNext(node: n);

    n = new Node();
    n.SetValue(30);
    head.GetNext().SetNext(node: n);
}
```

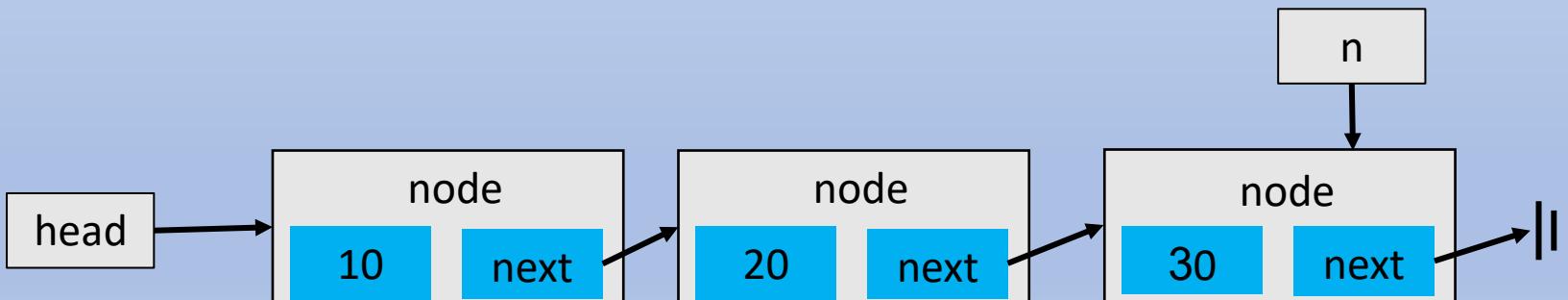




מה תדפיס התוכנית הבאה ?

```
public static void AddNodes ()  
{  
    Node head = new Node();  
    head.SetValue(10);  
  
    Node n = new Node();  
    n.SetValue(20);  
    head.SetNext(node: n);  
  
    n = new Node();  
    n.SetValue(30);  
    head.GetNext().SetNext(node: n);  
  
    Console.WriteLine(value: head.GetValue());  
    Console.WriteLine(value: head.GetNext().GetValue());  
    Console.WriteLine(value: head.GetNext().GetNext().GetValue());  
}
```

```
10  
20  
30  
Press any key to continue . . .
```

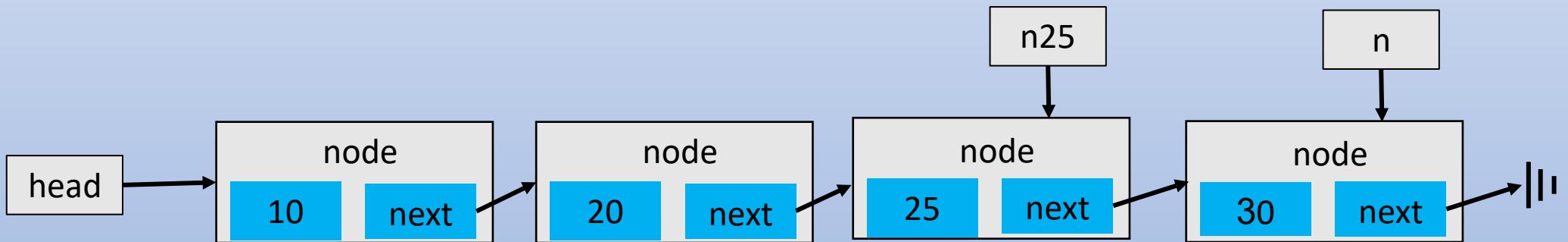




כיצד נוסיף חוליה 25 למקוםה ?



```
Node n25 = new Node();
n25.SetValue(25);
head.GetNext().SetNext(node: n25);
n25.SetNext(node: n);
```



```
Console.WriteLine(value: head.GetValue());
Console.WriteLine(value: head.GetNext().GetValue());
Console.WriteLine(value: head.GetNext().GetNext().GetValue());
Console.WriteLine(value: head.GetNext().GetNext().GetNext().GetValue());
```

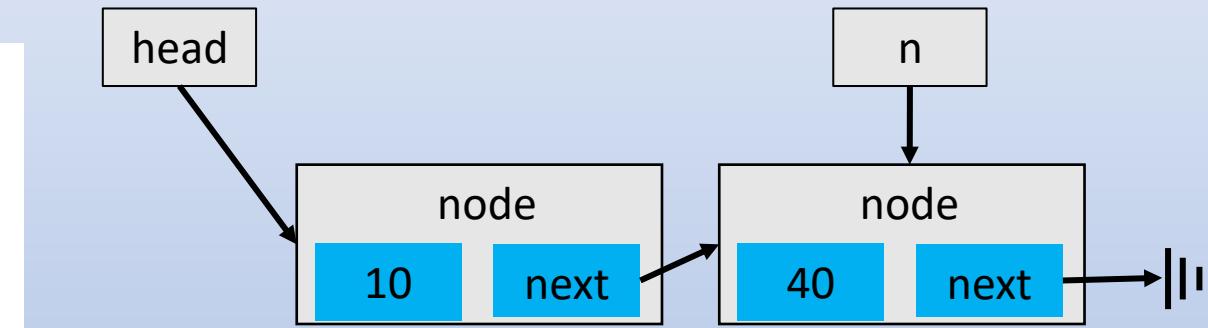
```
10
20
25
30
Press any key to continue . . .
```



מהי תמונה הזיכרון של הקוד הבא?

```
references
public static void loop()
{
    Node head = new Node();
    head.SetValue(10);
    Node n;

    for (int i=2; i < 5; i++)
    {
        n = new Node();
        n.SetValue(i * 10);
        head.SetNext(node: n);
    }
}
```



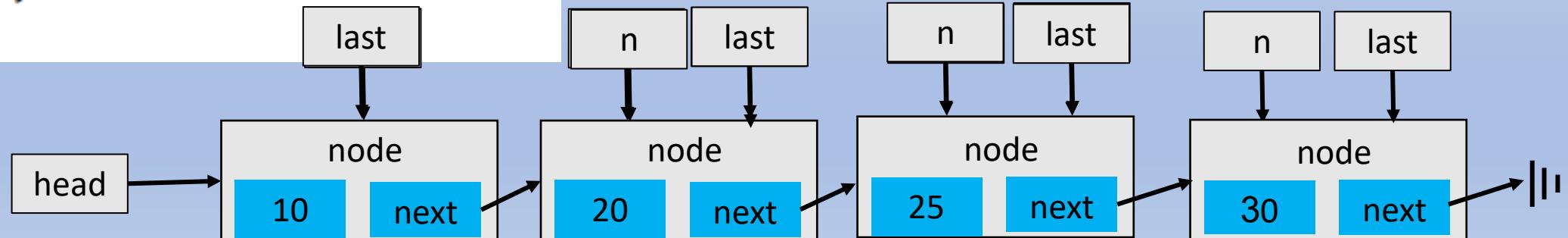
- החוליות (30, 20, 10) בוטלו כיון שאין אליהן כל מצביע.
- ה-Garbage Collector של השפה שחרר את הזיכרון בהם הם נשמרו, והם אבדו.



שאלה – כתוב פעולה שמשרשרת 10 חוליות עם הערכים ... 10,20,30 ...

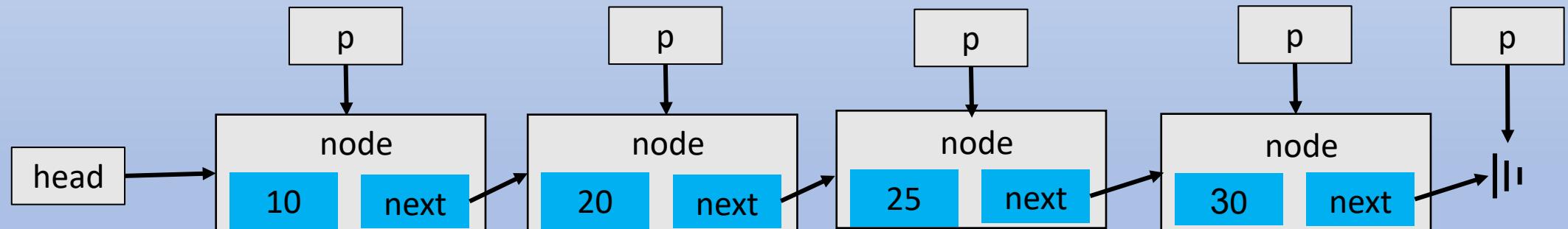
```
public static void loop2()
{
    Node head = new Node();
    head.SetValue(10);
    Node last = head;

    for (int i = 2; i <= 10; i++)
    {
        Node n = new Node();
        n.SetValue(i * 10);
        last.SetNext(node: n);
        last = n;
    }
}
```



תרגיל – כתוב פעולה המקבלת מצביע לראש השרשרת ומדפיסה את כל הערכים

```
1 reference
public static void print(Node p)
{
    while (p != null)
    {
        Console.WriteLine(value: p.GetValue());
        p = p.GetNext();
    }
}
```



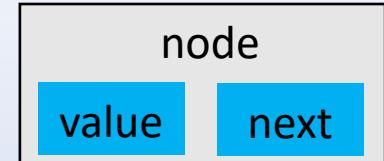
תרגיל – כתוב פעולה המקבלת מערך של מספרים שלמים ויצרת ממנו רשימה חוליות

```
public static Node createList (int[] arr)
```

```
public static Node createList (int[] arr) {  
    Node head = new Node(0);  
    Node tail = head;  
    for (int i=0; i < arr.length; i++) {  
        Node n = new Node(arr[i]);  
        tail.SetNext(n);  
        tail = tail.GetNext();  
    }  
    return head.GetNext();  
}
```



החוליה הגנרטיבית



- נותרה לנו בעיה שהחוליה שלנו מוגבלת ל `to` בלבד.
- אמנם אנחנו יכולים לבנות מחלוקת חדשה לכל סוג ערך, אבל זה לא יעיל.
- **פתרון:**
 - ניתן להגדיר את הערך כ `Object`.
 - אפשר יהיה להזין לחוליות ערכים מסוגים שונים.
 - החסרון - נדרש לבצע כל הזמן המרות כדי להשתמש באובייקטים.
- **חוליה גנרטיבית** – שפת `#c` מאפשרת לנו ליצור מחלוקות גנרטיביות, כלומר מחלוקות שבעת הקומpileציה נגדיר את סוג המשתניםuai them הם יעבדו.



מחלקה גנריית

- מחלקה גנריית היא מחלקה שבעת יצרת האובייקט אנחנו מעבירים לה את סוג המשתנה (int, double, car) ובאמצעות סוג מגדירים את סוג המשתנים.

```
public class GenericClass <T, U>
{
    private T firstProp;
    private U secondProp;

    2 references
    public GenericClass (T first, U second){
        this.firstProp = first;
        this.secondProp = second;
    }

    1 reference
    public void setFirstProp (T first) { this.firstProp = first; }

    0 references
    public T getFirstProp () { return this.firstProp; }

    3 references
    public U getSecondProp () { return this.secondProp; }

    0 references
    public void setSecondProp (U second) { this.secondProp = second; }
}
```

```
static void Main(string[] args)
{
    GenericClass<int, string> contact = new GenericClass<int, string>(first: 3, second: "Gilad");
    string name = contact.getSecondProp();
    contact.setFirstProp(first: 25);

    GenericClass<int, int> point = new GenericClass<int, int>(first: 3, second: 5);
    int x = point.getSecondProp();
    int y = point.getSecondProp();
}
```



קרית חינוך "פארק המדע"

בית לערבים, למציאות וחדשנות

```
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(); }
}
```

החוליה הగנרטיבית

החוליה הגנרטיבית Node<T>

תיאור הפעולה	חתימת הפעולה
בנאים:	
פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה ערך null	Node (T x)
פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה הערך next (אך המועבר כפרמטר יכול להיות גם null)	Node (T x , Node<T> next)
שאלות:	
אם T מחלק עוטפת לטיפוס בסיסי (Integer, Double, Character) ווחזר ערך החוליה, ואם הפניה לעצם, תוחזר הפניה לעצם זה	T GetValue ()
מחזרת הפניה לחוליה הבא	Node<T> GetNext ()
פעולה המחזיר את אמת אםiaz מהנה לחוליה נוספת (כלומר אין null) ושקר אחרה	bool HasNext ()
פעולה המחוירה מהירות המתארת את מצב העצם	string ToString ()
(*) סימוכיות: אם T עצם מטיפוס פשוט ← O(1) ואם T מייצג אוסף ← O(T) או רק האוסף כפונקציה של n	
פוקוות:	
הפעולה משנה את ערך ה- value של החוליה ל- x	void SetValue (T x)
הפעולה משנה את ערכו של next להיות next חדש (ערך null המתקבל כפרמטר יכול להיות גם null)	void SetNext (Node<T> next)



שימוש בחוליה הגנרטיבית

- ניצור שרשרת של 10 חוליות גנריות (עם ערך מספר שלם ... 30, 20, 10):

```
public static void useGenericNode ()  
{  
    Node<int> head = new Node<int>(value: 10);  
    Node<int> p = head;  
    for (int i=2; i<=10; i++)  
    {  
        p.SetNext(new Node<int>(value: i * 10));  
        p = p.GetNext();  
    }  
}
```

- שימושו לה שבעת יצירת החוליה אנחנו חיבים להחליף את `z` בסוג הערך שהחרנו.
- הפקודה `new` מחייבת מצביע לאובייקט. על כן, ניתן להשתמש בה כך שהמצביע נשמר ישירות ב `next`. ניתן היה לכתוב פקודה זו בשתי שורות:

```
Node<int> n = new Node<int>(value: i * 10);  
p.SetNext(n);  
p = p.GetNext();
```



כתב את המחלקה

קרית חינוך "פרק המדע"

בית לערבים, למציאות וחדשנות

```
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(); }
}
```

החוליה הగורית Node<T>

תיאור הפעולה	חתימת הפעולה
בנאים:	
פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה ערך null	Node (T x)
פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה הערך next (אך next המועבר כפרמטר יכול להיות גם null)	Node (T x , Node<T> next)
שאלות:	
אם T מחלוקת עוטפת לטיפוס בסיסי (Integer, Double, Character) ווחזר ערך החוליה, ואם הפניה לעצם, תוחזר הפניה לעצם זה	T GetValue ()
מחזורת הפניה לחוליה הבא	Node<T> GetNext ()
פעולה המחזירה אמת אםiaz מהנה לחוליה נוספת (כלומר אין null) ושקר אחרת	bool HasNext ()
פעולה המוחזירה מחוירות המתארת את מצב העצם	string ToString ()
(*) סימוכיות: אם T עצם מטיפוס פשוט ← O(1) ואם T מייצג אוסף ← O(T) או רוח האוסף כפונקציה של n	
פוקוות:	
הפעולה משנה (מעדכנת) את ערך ה- value של החוליה ל- x	void SetValue (T x)
הפעולה משנה את ערכו של next להיות next חדש (ערך null המתקבל כפרמטר יכול להיות גם null)	void SetNext (Node<T> next)

תרגיל – כתוב פעולה המקבלת רשימה גנית ומדפיסה אותה

```
0 references
public static void printList (Node<int> head)
{
    while (head != null)
    {
        Console.WriteLine(value: head);
        head = head.GetNext();
    }
}
```

- הדף רשימה מתבצעת בולאה while המתחילה בראש הרשימה ומסתיימת כאשר מגאים ל null.
- שימוש לב כי בפעולה זו לא שמרנו על ראש הרשימה כיוון שנשאר עותק בפעולה המזמנת, ולא היינו צריכים לעשות פעולות נוספות על הרשימה.
- אם היינו צריכים לעשות פעולות נוספות על הרשימה בפעולה הנוכחית – חובה היה לשמר עותק של המצביע לראש הרשימה.



תרגיל 1 – פעולות בסיסיות על רשימה

- כתוב את המחלקה `<T> Note` בפרויקט שלך. פטור את התרגילים בעזרת מחלקה זו.

שם פעולה	תיאור
<code>CreateList(int [] arr)</code>	פעולה היוצרת רשימה מקוורת של החוליות שהתקבלו במערך ומחזירה את ראש הרשימה.
<code>PrintList (Node<T> head)</code>	הפעולה מקבלת מצביע לתחילת רשימה והוא מדפיסה את ערכי הרשימה בשורה אחת עם רווחים.
<code>Length(Node<T> head)</code>	הפעולה מקבלת מצביע לתחילת רשימה ומחזירה את מספר אברי הרשימה.
<code>GetNode(Node<T> head, int n)</code>	הפעולה מחזירה את האיבר במקומות <code>n</code>
<code>Add (Node<T> head, Node<T> node)</code>	הפעולה מקבלת מצביע לראש הרשימה, ומוסיפה את האיבר בתחילת הרשימה. מחזירה את ראש הרשימה החדש.
<code>Add (Node<T> head, int value, int n)</code>	הפעולה מקבלת מצביע לתחילת רשימה, ערך, ומקומות – ומוסיפה חוליה עם הערך במקומות המבוקש (יש להיזהר לא לנתק את המשך הרשימה)
<code>AddLast(Node<T> lst, Node<T> node)</code>	הפעולה מוסיפה את החוליה בסוף הרשימה.
<code>Pop (Node<T> head)</code>	הפעולה מוחקת את האיבר הראשון בראשיה ומחזירה את ראש הרשימה החדש.
<code>Del (Node<T> head, int n)</code>	הפעולה מוחקת את האיבר במקומות <code>n</code> (שים לב שלא לנתק את המשך שרשרת).
<code>Sum(Node<int> head)</code>	הפעולה מחזירה את סכום הערכים של הרשימה (ניתן להניח שהערכים ניתנים לסכימה).



תרגיל 2 – פועלות מתקדמות עם רשימות

תיאור	שם הפעולה
הפעולה משכפלת את הרשימה ומחזירה מצביע לרשימה חדשה.	CopyList (Node<string> head)
הפעולה מחזירה מצביע לרשימה חדשה בסדר הפוך (החוליה האחורונה היא הראשונה).	Reverse (Node<double> head)
הפעולה מוחפשת את האיבר הראשון עם הערך שהתקבל ומחזירה את המיקום שלו בראשימה.	Index (Node<int> head, int value)
הפעולה מוחפשת את האיבר הראשון עם הערך שהתקבל ומחזירה את האיבר (מצביע לאיבר).	Find (Node<string> head, string value)
הפעולה מחזירה את רשימה של הרשימה שהתקבלת מן האיבר start ועד לאיבר end. (אין צורך לבדוק תקינות האינדקסים).	Slice (Node<char> head, int start, int end)
הפעולה מקבלת שתי רשימות ובודקת אם הערכים שלהם זהים. אם כן, מחזירה אמת. אחרת שקר.	Equal(Node<int> L1, Node<int> L2)
הפעולה מקבלת שתי רשימות ומחזירה רשימה מאוחדת L2 אחרי L1. אין צורך להעתיק את האיברים (שימוש באיברים הקיימים).	Union(Node<double> L1, Node<double> L2)
הפעולה מקבלת שתי רשימות ומחזירה רשימה שבה האיברים בעלי הערך זהה.	Intersect(Node<string> L1, Node<string> L2)
הפעולה מוחקמת מן הרשימה את האיברים הכפולים ומשאירה רק איבר אחד מכל ערך.	Unique(Node<int> head)



המשך תרגיל 2 – פעולות מתקדמות עם רשימות

תיאור	שם הפעולה
הפעולהמחזירה את האיבר עם הערך הגדול ביותר. אתם יכולים להניח כי ניתן להשוות בין הערכים באמצעות <code><</code> .	Max(Node<int> head)
הפעולה מקבלת שתי רשימות ממוגנות (אתם יכולים להניח כי ניתן להשוות בין הערכים באמצעות <code><</code> או <code>></code>). הפעולהמחזירה רשימה ממוגנת וממוגנת (אין להעתק איברים).	UnionSort(Node<double> L1, Node<double> L2)
הפעולה מקבלת רשימה של <code>int</code> שהערכים בין 1 ל- 100. הפעולה תחזיר את המספר השכיח ביותר (שימוש הכי הרבה פעמים). ניתן להשתמש במבנה נתונים נוסף.	commonValue (Node<int> head)
הפעולה מקבלת רשימה ומחזירה אמת היא ערכי האיברים עולים ממש או יורדים ממש. אתם יכולים להניח כי ניתן לבצע השוואה בין האיברים באמצעות <code><</code> או <code>></code> .	Consistent(Node<int> head)
הפעולה מקבלת רשימה ומחזירה true אם היא פולינדרום. רמז: ניתן להשתמש בשתי פעולות קודמות שכתבתם והתשובה מסכמת בשלוש שורות.	isPalindrome(Node<char> head)
הפעולה מקבלת שתי רשימות ומחזירה אמת אם רשימה 2 מוכלת ב 1. $L_1 \subseteq L_2$	Contains(Node<int> L1, Node<int> L2)
הפעולה מקבלת שתי רשימות ומאחדת אותן לרשימה אחת בצורה ריצ'רצ' (איבר מ 1, איבר מ 2 ו חוזר חלילה).	Zipper (Node<string> L1, Node<string> L2)
פעולה מקבלת רשימה של מספרים ממשיים, יוצרת רשימה של המספרים החיובים מתוך הרשימה, מדפסה (באמצעות <code>PrintList()</code>) את הרשימה החדשה ולאחריה את הרשימה המקורי, ומחזירה את הרשימה החדשה.	Positive(Node<double>)



תרגילים מבגרויות

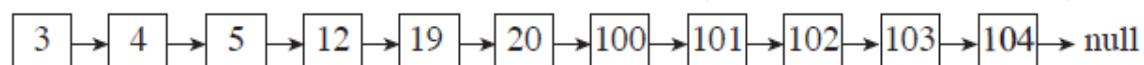
תיאור	שם הפעולה
<p>הפעולה מקבלת שרשרת חוליות של מספרים שלמים ומחזירה שרשרת חדשה כזו: עבור כל תת-רשימה של מספרים עולים ב 1, שבה כל מספר גדול ממש מוקודמו, יופיע ברשימה החדשה איבר אחד עם סכום תת הסדרה. כל תת סדרה מסתייםת כאשר אחר האיבר האחרון מופיע איבר שווה או קטן לו.</p>	<code>SumOfsubList(Node<int> L)</code>
<p>רשימה L תיקרא מושולשת אם היא מקיימת את שלושת התנאים הבאים: (א) הראשינה אינה ריקה; (ב) מספר האיברים בה מחלק ב-3; (ג) האיברים בשליש הראשון של הרשימה מכילים את אותם ערכים שמכילים האיברים בשליש השני ושליש השלישי ובאותו הסדר. כתוב פועלה המקבלת רשימה ומחזירה אמת אם היא מושולשת אחרת שקר.</p> 	<code>isTripleList (Node<int> L)</code>
<p>הפעולה מקבלת רשימה ממוגנת של int ומוסיפה איבר עם הערך החדש במקום המתאים. אם הערך קיים ניתן להוסיף לפניו או אחרי החוליה השווה לו.</p>	<code>addSort(Node<int> L, int n)</code>
<p>הפעולה מקבלת רשימה של מספרים שלמים ומחזירה רשימה חדשה ממוגנת (הרשימה המקורית לא משתנה). ניתן וכדי לעשות שימוש בפעולות קודמות.</p>	<code>sortList(Node<int> L)</code>

שאלות מבחן 2010

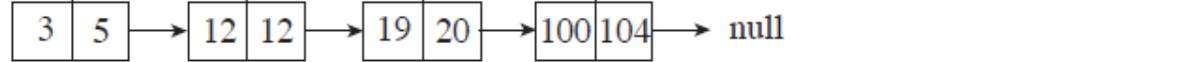
ראה למטה חתימת הפעולה המבוקשת

- ל. 2. **T** היא רשימה המכילה מספרים שלמים שונים זה מזה וממונינים בסדר עולה.
רשימת הטוחחים של T היא רשימה חדשה שנבנית באופן הזה: בעבור בל רצף של
 מספרים עוקבים ב- T יהיה **ברשימה הטוחחים 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 רצף הוא באורך 1, הוא מיוצג
ברשימה הטוחחים על ידי איבר שני המספרים בו שוים.

לדוגמה, בעבור הרשימה T שלפניך:



רשימת הטווחים של T תהיה:



לפניך תיאור חלקו של המחלקה `RangeNode`, המייצגת איבר בראשימת הטעושים.

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

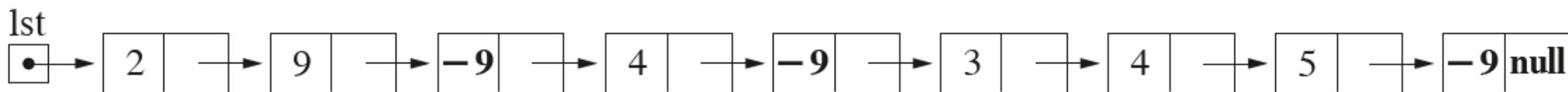
ממש ב- Java או ב- #C פעולה חיצונית שתקבל רשימה לא ריקה, המכילה מספרים שלמים שונים זה מזה וממוינים בסדר עולה, ותחזיר את **רשימת התווים** שלה.

```
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 הגדל מ- 0 .
• "שרשרת ספרות" היא שרשרת חוליות שכל חוליה בה מכילה ספרה בין 0 ל- 9 (כולל) או את המספר (9 –).
כל רצף ספרות בשרשרת 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 בשרשרת.

לפניך דוגמה ל"שרשרת ספרות" המייצגת את המספרים: 92 , 4 , 543 .



כתב פועלה חיצונית בשפת Java או בשפת C# buildDigit باسم , BuildDigit המתקבלת הפניה lst
שאינה null ל"שרשרת מספרים שלמים חיוביים". הפועלה תחזיר "שרשרת ספרות" המייצגת את המספרים
שב"שרשרת מספרים שלמים חיוביים" לפי הסדר.

שאלה בرمת בגרות

ממשו את הפעולה הבאה ונתחו את יעילותה :

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

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

דוגמאות :

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



קורסיה בראשימות

תיאור	שם הפעולה
כתבו פעולה רקורסיבית המדפיסה את הרשימה מהסוף להתחלה.	RecursionPrint(Node<int> head)
כתבו פעולה רקורסיבית הסופרת את מספר האיברים בראשימה.	RecursionLength(Node<double> head)
כתב פועלה רקורסיבית הסופרת את מספר הפעמים שהערך מופיע בראשימה.	RecursionCount (Node<int> head, int value)
כתבו פעולה רקורסיבית המחזיר את הערך המקסימלי.	RecursionMax (Node<int> lst)
כתב פועלה רקורסיבית המחזיר עותק של הרשימה.	RecursionCopy (Node<string> head)
כתב פועלה רקורסיבית המחזיר עותק של הרשימה בסדר הפוך.	RecursionReverse (Node<string> head, Node<string> newList)