



קריית חינוך "פארק המדע"

בית לערכים, למצוינות וחדשנות



חוליה - שרשרת חוליות רשימה מקושרת

כיתה יא

גלעד מרקמן



קריית חינוך "פארק המדע"

בית לערכים, למצוינות וחדשנות

הצורך במבנה נתונים דינאמי

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

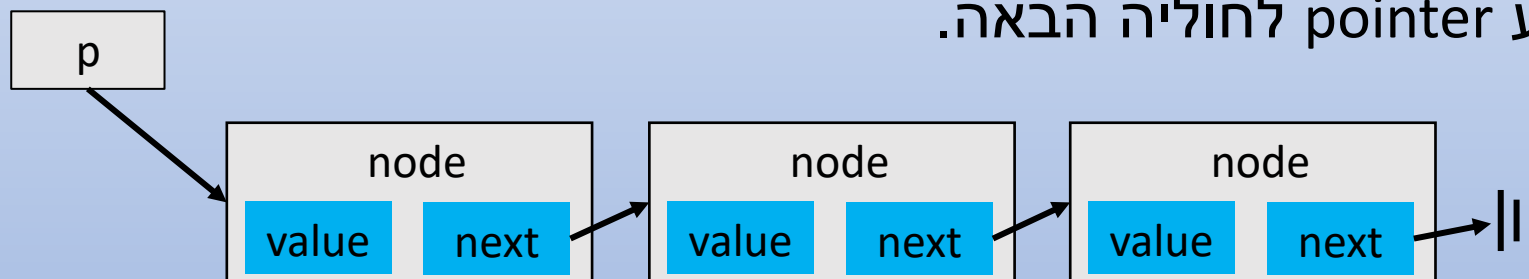


קריית חינוך "פארק המדע"

בית לערכים, למצוינות וחדשנות

החוליה Node

- החוליה היא אבן הבניין של מבני נתונים דינאמיים 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, להוסיף, ולהוריד חוליות בהתאם לצורך.



חוליה פשוטה

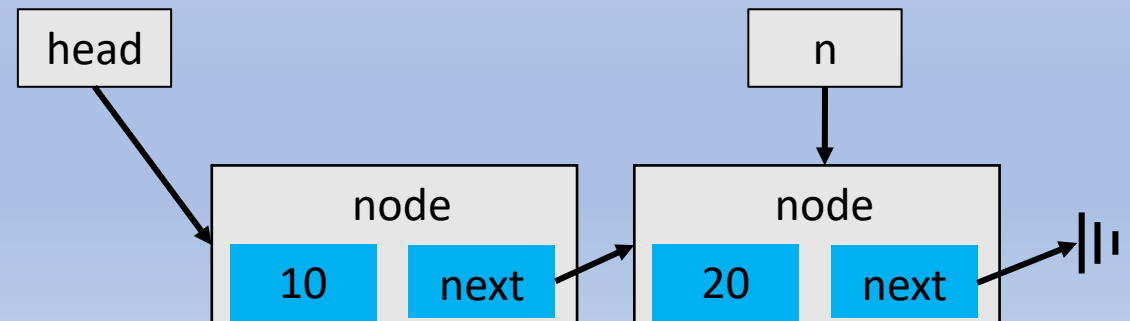
- נגדיר לדוגמה חוליה המקבלת ערכים של int, ונבנה שרשרת של שתי חוליות:

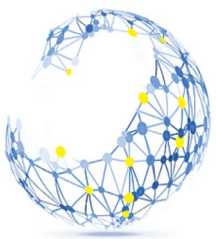
```
class Node
{
    private int value;
    private Node next;

    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);
}
```



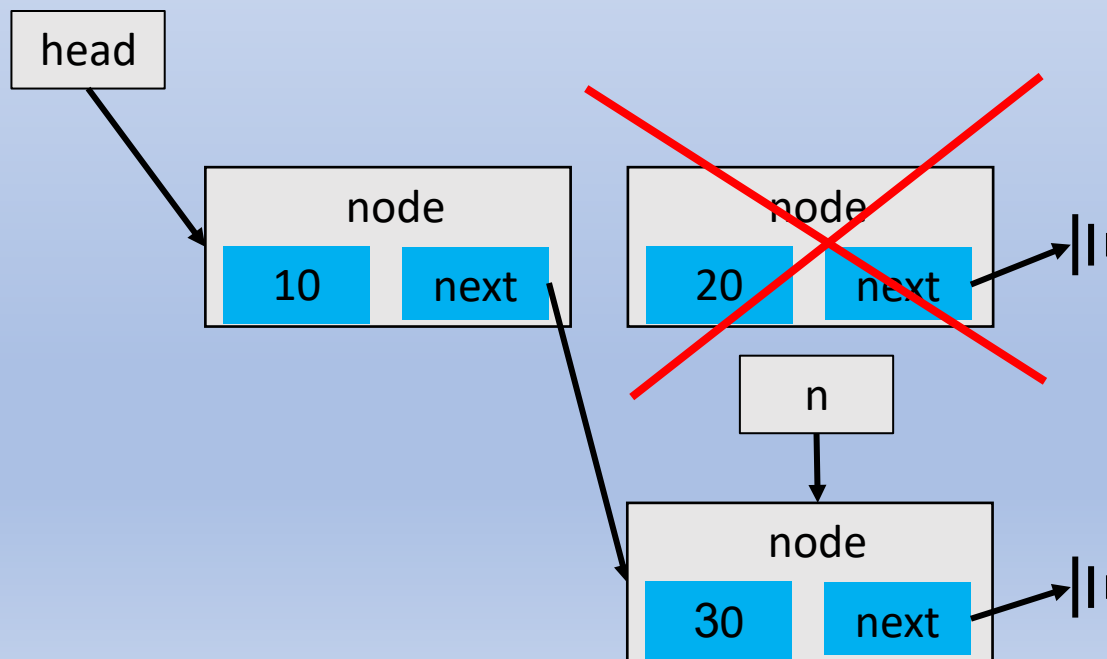
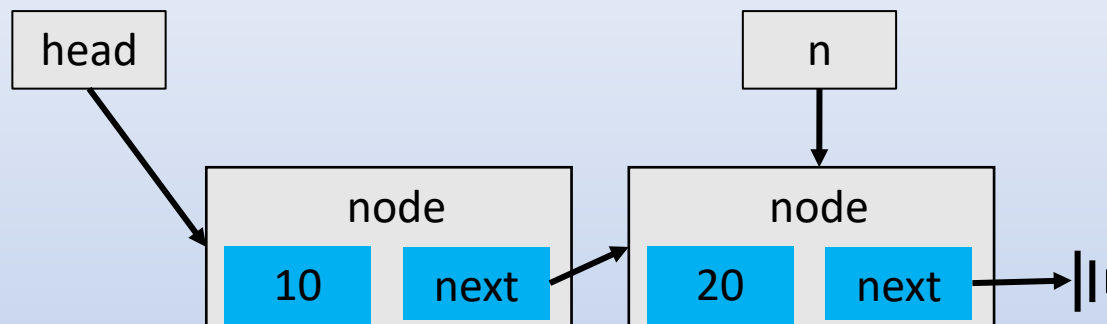


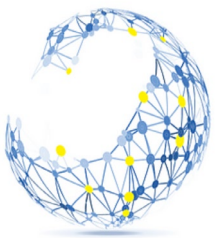
מהי תמונת הזיכרון של הקוד הבא?

```
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);
}
```



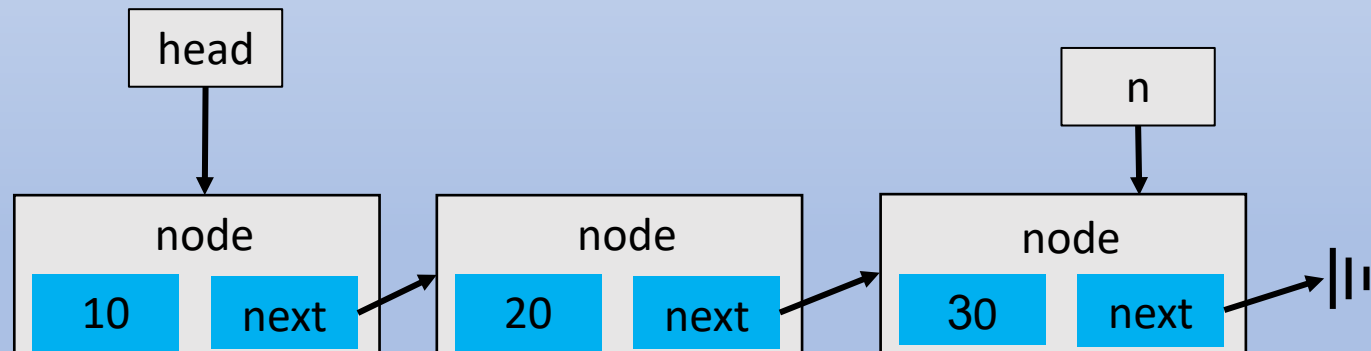
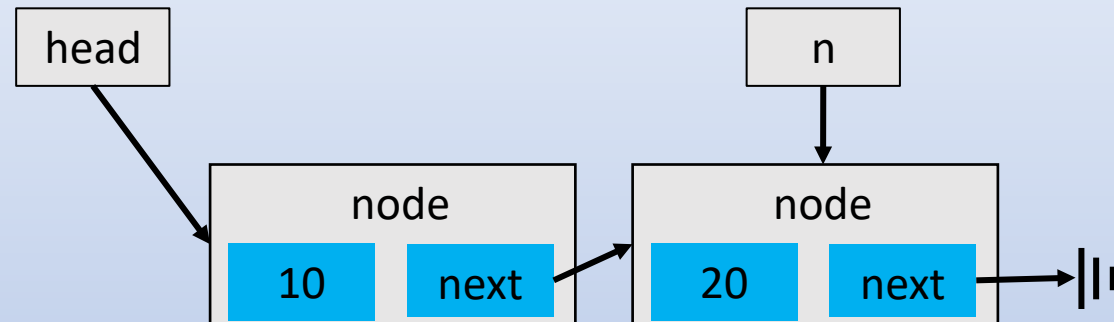


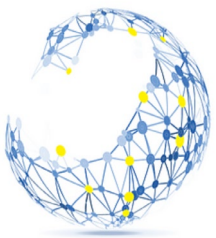
מהי תמונת הזכרון של הקוד הבא

```
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 ?

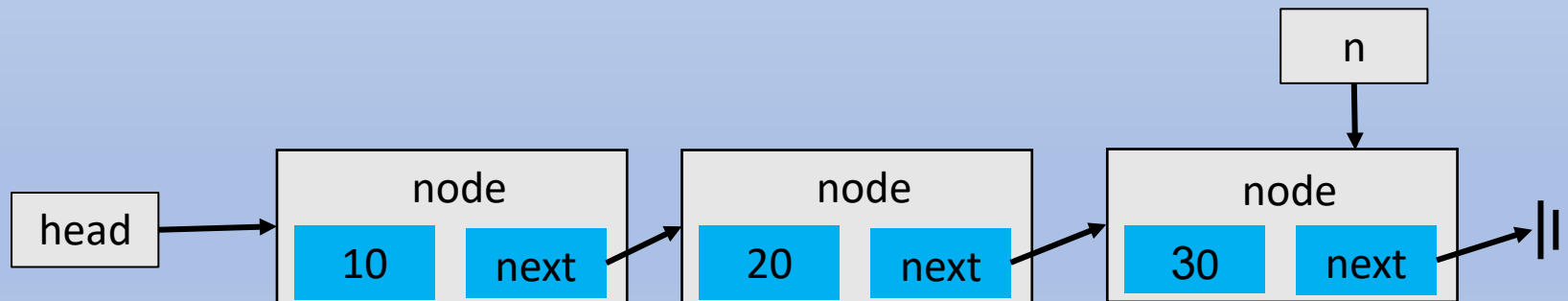
```
public static void AddNodes ()
{
    Node head = new Node();
    head.SetValue(10);

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

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

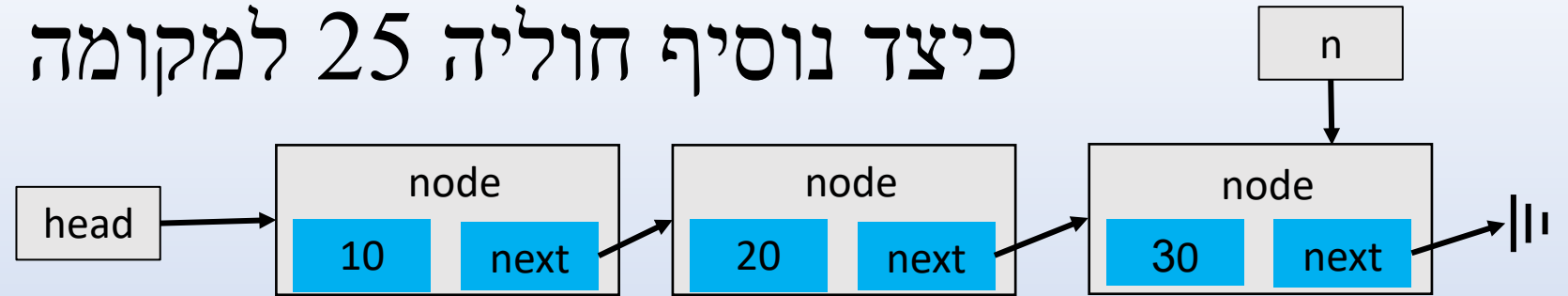
    Console.WriteLine(value: head.GetValue());
    Console.WriteLine(value: head.GetNext().GetValue());
    Console.WriteLine(value: head.GetNext().GetNext().GetValue());
}
```

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

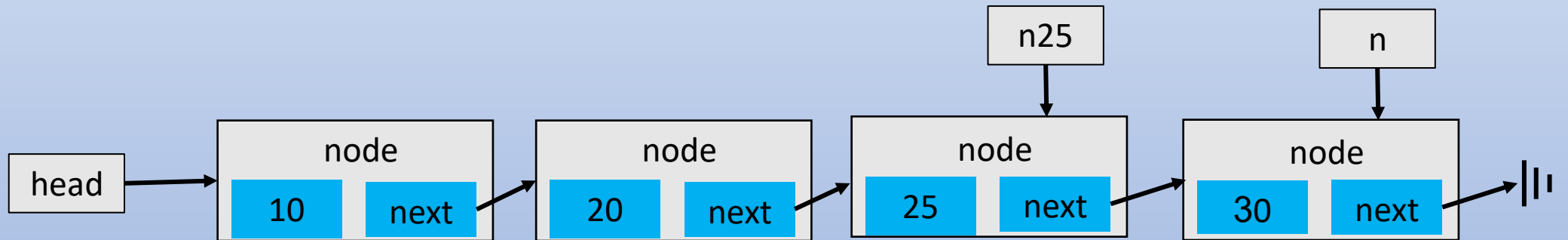




כיצד נוסיף חוליה 25 למקומה ?



```
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());
```

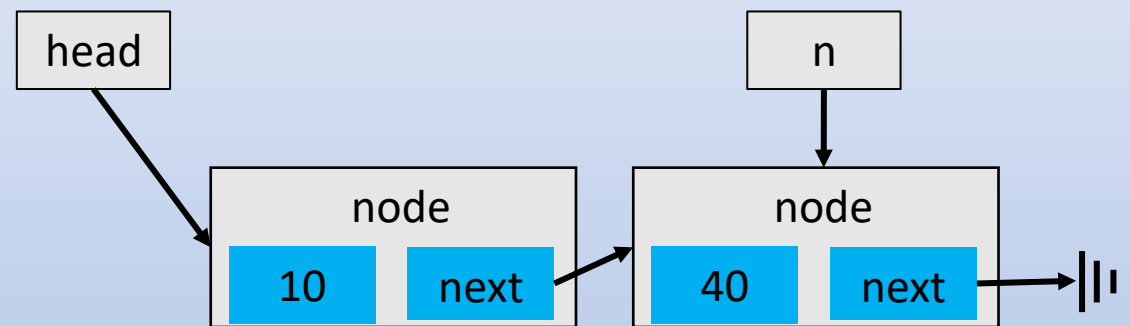
```
10  
20  
25  
30  
Press any key to continue . . .
```




מהי תמונת הזכרון של הקוד הבא?

```
0 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);
    }
}
```



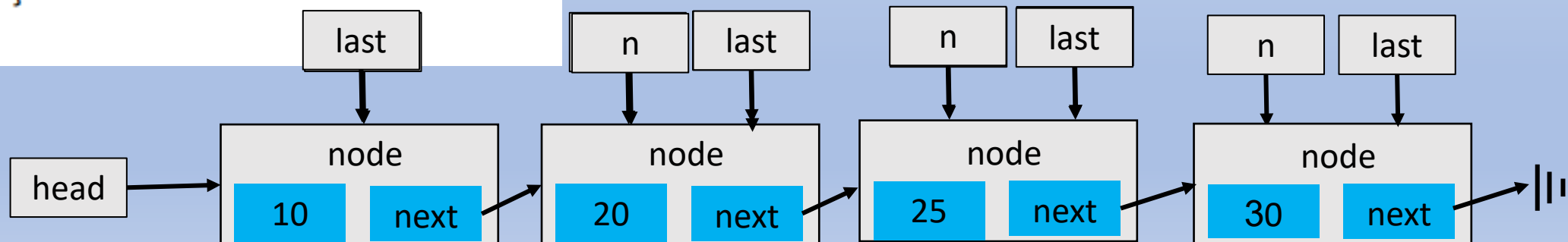
- החוליות $n(20)$, $n(30)$ בוטלו כיוון שאין אליהן כל מצביע.
- ה Garbage Collector של השפה שחרר את הזכרון בהם הם נשמרו, והם אבדו.



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

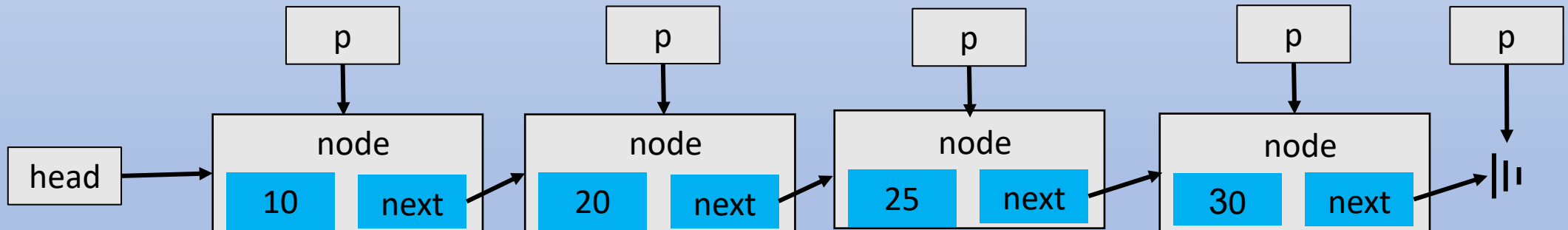
```
public static void loop2()
{
    Node head = new Node();
    head.SetValue(10);
    Node last = head;

    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 רשימת חוליות

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

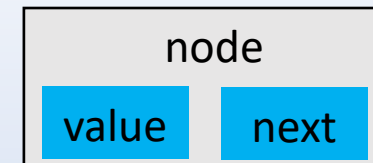
```
public static Node createList (int[] arr) {  
    Node head = new Node(0);  
    Node tail = head;  
    for (int i=0; i < arr.length; i++) {  
        Node n = new Node(arr[i]);  
        tail.SetNext(n);  
        tail = tail.GetNext();  
    }  
    return head.GetNext();  
}
```



קריית חינוך "פארק המדע"

בית לערכים, למצוינות וחדשנות

החוליה הגנרית



- נותרה לנו בעיה שהחוליה שלנו מוגבלת ל `int` בלבד.
- אמנם אנחנו יכולים לבנות מחלקה חדשה לכל סוג ערך, אבל זה לא יעיל.
- פתרון:
 - ניתן להגדיר את הערך כ `Object`.
 - אפשר יהיה להזין לחוליות ערכים מסוגים שונים.
 - החסרון - נצטרך לבצע כל הזמן המרות כדי להשתמש באובייקטים.
- החוליה גנרית – שפת `c#` מאפשרת לנו ליצור מחלקות גנריות, כלומר מחלקות שבעת הקומפילציה נגדיר את סוג המשתנים שאיתן הם יעבדו.



מחלקה גנרית

- מחלקה גנרית היא מחלקה שבעת יצירת האובייקט אנחנו מעבירים לה את סוג משתנה (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; }

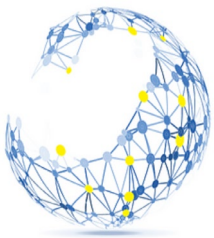
    0 references
    public T getFirstProp () { return this.firstProp; }

    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();
}
```



החוליה הגנרית Node<T>

קריית חינוך "פארק המדע"

בית לערכים, למצוינות 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(); }
}
```

| Node<T> הגנרית | |
|-----------------------------|---|
| תיאור הפעולה | חתימות הפעולה |
| בנאים: | |
| Node (T x) | פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה הערך null |
| Node (T x , Node<T> next) | פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה הערך next (ערך המועבר כפרמטר יכול להיות גם null) |
| שאלות: | |
| T GetValue () | אם T מחלקה עוטפת לטיפוס בסיסי (Integer, Double, Character) יוחזר ערך החוליה, ואם הפניה לעצם, תוחזר הפנייה לעצם זה |
| Node<T> GetNext () | מחזרת הפנייה לחוליה הבאה |
| bool HasNext () | פעולה המחזירה אמת אם next מפנה לחוליה נוספת (כלומר אינו null) ושקר אחרת |
| string ToString () | פעולה המחזירה מחרוזת המתארת את מצב העצם (*) סיבוכיות: אם T עצם מטיפוס פשוט $\leftarrow O(1)$ ואם T מייצג אוסף $\leftarrow O(T)$ (אורך האוסף כפונקציה של n) |
| פקודות: | |
| void SetValue (T x) | הפעולה משנה (מעדכנת) את ערך ה- value של החוליה ל- x |
| void SetNext (Node<T> next) | הפעולה משנה את ערכו של next להיות next חדש (ערך next המתקבל כפרמטר יכול להיות גם null) |



שימוש בחוליה הגנרית

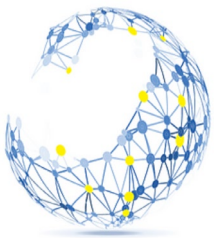
- ניצור שרשרת של 10 חוליות גנריות (עם ערך מספר שלם ... 10,20,30):

```
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();
    }
}
```

- שימו לה שבעת יצירת החוליה אנחנו חייבים להחליף את T בסוג הערך שבחרנו.

- הפקודה new מחזירה 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();
```

כתוב את המחלקה Node<T>

קריית חינוך "פארק המדע"
בית לערכים, למצוינות וחדשנות

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

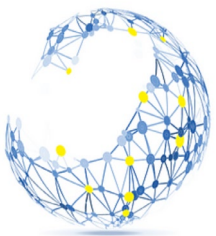
    2 references
    public Node(T value) { this.value = value; this.next = null;}
    0 references
    public Node(T value, Node<T> next) { this.value = value; this.next = next;}
    1 reference
    public T GetValue() { return this.value; }
    0 references
    public void SetValue(T value) { this.value = 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(); }
}
```

| Node<T> הגנרית | |
|-----------------------------|---|
| תיאור הפעולה | חתימות הפעולה |
| בנאים: | |
| Node (T x) | פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה הערך null |
| Node (T x , Node<T> next) | פעולה הבונה חוליה שבערך value שלה יהיה x וב- next שלה יהיה הערך next (ערך המועבר כפרמטר יכול להיות גם null) |
| שאלות: | |
| T GetValue () | אם T מחלקה עוטפת לטיפוס בסיסי (Integer, Double, Character) יוחזר ערך החוליה, ואם הפניה לעצם, תוחזר הפנייה לעצם זה |
| Node<T> GetNext () | מחזרת הפנייה לחוליה הבאה |
| bool HasNext () | פעולה המחזירה אמת אם next מפנה לחוליה נוספת (כלומר אינו null) ושקר אחרת |
| string ToString () | פעולה המחזירה מחרוזת המתארת את מצב העצם (*) סיבוכיות: אם T עצם מטיפוס פשוט $O(1)$ ואם T מייצג אוסף $O(T)$ (אורך האוסף כפונקציה של n) |
| פקודות: | |
| void SetValue (T x) | הפעולה משנה (מעדכנת) את ערך ה- value של החוליה ל- x |
| void SetNext (Node<T> next) | הפעולה משנה את ערכו של next להיות next חדש (ערך next המתקבל כפרמטר יכול להיות גם null) |

תרגיל – כתוב פעולה המקבלת רשימה גנרית ומדפיסה אותה

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

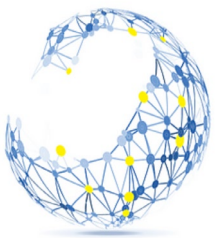
- הדפסת רשימה מתבצעת בלולאה while המתחילה בראש הרשימה ומסתיימת כאשר מגיעים ל null.
- שימו לב כי בפעולה זו לא שמרנו על ראש הרשימה כיוון שנשאר עותק בפעולה המזמנת, ולא היינו צריכים לעשות פעולות נוספות על הרשימה.
- אם היינו צריכים לעשות פעולות נוספות על הרשימה בפעולה הנוכחית – חובה היה לשמור עותק של המצביע לראש הרשימה.



תרגיל 1 – פעולות בסיסיות על רשימה

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

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



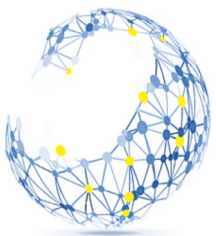
תרגיל 2 – פעולות מתקדמות עם רשימות

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

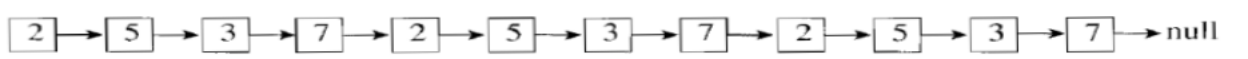


המשך תרגיל 2 – פעולות מתקדמות עם רשימות

| שם הפעולה | תיאור |
|---|--|
| Max(Node<int> head) | הפעולה מחזירה את האיבר עם הערך הגדול ביותר. אתם יכולים להניח כי ניתן להשוות בין הערכים באמצעות <. |
| UnionSort(Node<double> L1, Node<double> L2) | הפעולה מקבלת שתי רשימות ממוינות (אתם יכולים להניח כי ניתן להשוות בין הערכים באמצעות < או >). הפעולה מחזירה רשימה ממוזגת וממוינת (אין להעתיק איברים). |
| commonValue (Node<int> head) | 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. |
| Consistent(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 אמצעות < או >). |
| isPalidrome(Node<char> head) | הפעולה מקבלת רשימה ומחזירה true אם היא פולינדרום. רמז: ניתן להשתמש בשתי פעולות קודמות שכתבתם והתשובה מסכמת בשלוש שורות. |
| Contains(Node<int> L1, Node<int> L2) | הפעולה מקבלת שתי רשימות ומחזירה אמת אם רשימה L2 מוכלת ב L1. $L2 \subseteq L1$. |
| Zipper (Node<string> L1, Node<string> L2) | הפעולה מקבלת שתי רשימות ומאחדת אותן לרשימה אחת בצורת ריצ'רצ (איבר מ L1, איבר מ L2 וחוזר חלילה). |
| Positive(Node<double>) | פעולה המקבלת רשימה של מספרים ממשיים, יוצרת רשימה של המספרים החיוביים מתוך הרשימה, מדפיסה (באמצעות PrintList()) את הרשימה החדשה ולאחריה את הרשימה המקורית, ומחזירה את הרשימה החדשה. |



תרגילים מבגרויות

| שם הפעולה | תיאור |
|-----------------------------|--|
| SumOfsubList(Node<int> L) | 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 מסתיימת כאשר אחר האיבר האחרון מופיע איבר שווה או קטן לו. |
| isTripleList (Node<int> L) | רשימה L תיקרא משולשת אם היא מקיימת את שלושת התנאים הבאים: (א) הרשימה אינה ריקה; (ב) מספר האיברים בה מתחלק ב-3; (ג) האיברים בשליש הראשון של הרשימה מכילים את אותם ערכים שמכילים האיברים בשליש השני ובשליש השלישי ובאותו הסדר. כתוב פעולה המקבלת רשימה ומחזירה אמת אם היא משולשת אחרת שקר.  |
| addSort(Node<int> L, int n) | הפעולה מקבלת רשימה ממוינת של int ומוסיפה איבר עם הערך החדש במקום המתאים. אם הערך קיים ניתן להוסיפו לפני או אחרי החוליה השווה לו. |
| sortList(Node<int> L) | הפעולה מקבלת רשימה של מספרים שלמים ומחזירה רשימה חדשה ממוינת (הרשימה המקורית לא משתנה). ניתן וכדאי לעשות שימוש בפעולות קודמות. |
| | |

שאלה מבגרות 2010

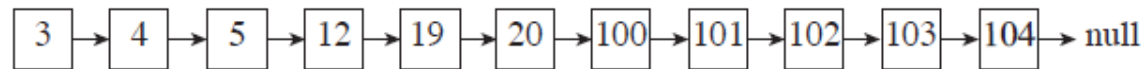
ראה למטה חתימת הפעולה המבוקשת

2.

L היא רשימה 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 איבר ששני המספרים בו שווים.

לדוגמה, בעבור הרשימה L שלפניך:



רשימת הטווחים של L תהיה:



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

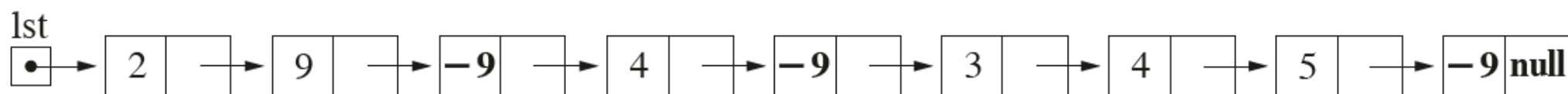
| RangeNode | |
|------------------------------------|---------------------------|
| private int from; | // המספר הקטן ביותר ברצף |
| private int to; | // המספר הגדול ביותר ברצף |
| public RangeNode(int from, int to) | |

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



כתוב פעולה חיצונית בשפת Java בשם `buildDigit` או בשפת C# בשם `BuildDigit`, המקבלת 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 בגרות

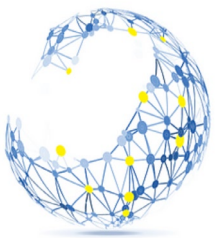
ממשו את הפעולה הבאה ונתחו את יעילותה :

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

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

דוגמאות :

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



רקורסיה ברשימות

| שם הפעולה | תיאור |
|---|--|
| RecursionPrint(Node<int> head) | כתבו פעולה רקורסיבית המדפיסה את הרשימה מהסוף להתחלה. |
| RecursionLength(Node<double> head) | כתבו פעולה רקורסיבית הסופרת את מספר האיברים ברשימה. |
| RecursionCount (Node<int> head, int value) | כתוב פעולה רקורסיבית הסופרת את מספר הפעמים שהערך מופיע ברשימה. |
| RecursionMax (Node<int> lst) | כתבו פעולה רקורסיבית המחזירה את הערך המקסימלי. |
| RecursionCopy (Node<string> head) | כתוב פעולה רקורסיבית המחזירה עותק של הרשימה. |
| RecursionReverse (Node<string> head, Node<string> newLst) | כתוב פעולה רקורסיבית המחזירה עותק של הרשימה בסדר הפוך. |
| | |
| | |