חומרים שהוכנו על-ידי משתתפי קורס מורים מובילים תשע"ה

ניתן להשתמש בחומרים לצורך הוראה בלבד.

לא ניתן לפרסם את החומרים או לעשות בהם כל שימוש מסחרי

ללא קבלת אישור מראש מצוות הפיתוח

כתיבה ועריכה:

אביטל גרינולד, דורית כהן, משה שטיינר

פעולות בנייה

```
static Random rnd = new Random();
static Scanner in = new Scanner(System.in);
```

1. פעולה המקבלת מספר שלם, n, ומחזירה רשימה של שלמים מסודרים מ-1 עד n

```
public static Node<int> Create1_N(int n)
{
    Node<int> list = null;
    for (; n > 0; n--)
        list = new Node<int>(n, list);
    return list;
}
```

2. פעולה המקבלת רשימה של שלמים ומדפיסה אותם עם "->" מפרידים, ובסוף null
רשימה ריקה מדפיסה null

```
public static void Print(Node<int> list)
{
    if (list == null)
        Console.WriteLine("null");
    else
    {
        Console.Write(list.GetValue() + "->");
        Print(list.GetNext());
    }
}
```

2. אותה פעולת הדפסה, באופן איטרטיבי

```
public static void Print1(Node<int> list)
{
    if (list == null)
        Console.WriteLine("null");
    else
    {
        while (list != null)
        {
            Console.Write(list.GetValue() + "->");
            list = list.GetNext();
        }
        Console.WriteLine("null");
    }
}
```

3. פעולה המקבלת מספר שלם n, וטווח של שלמים מ-low ל-high, הפעולה מחזירה רשימה אקראית של שלמים באורך n, ובטווח הנתון

```
public static Node<int> BuildRandom(int n, int low, int high)
{
    Node<int> list = null;
    for (; n > 0; n--)
        list = new Node<int>(low + rnd.Next(high - low), list);
    return list;
}
```

4. פעולה הבונה ומחזירה רשימה של שלמים המתקבלים מהמשתמש, עד לקליטת המספר 99

```
public static Node<int> Create()
{
    Node<int> list = null, last = null;
    Console.Write("Enter a number, 99 to quit:  ");
    int n = int.Parse(Console.ReadLine());
    while (n != 99)
    {
        if (last == null)
        {
            last = new Node<int>(n);
            list = last;
        }
        else
        {
            last.SetNext(new Node<int>(n));
            last = last.GetNext();
        }
        Console.Write("Enter a number, 99 to quit:  ");
        n = int.Parse(Console.ReadLine());
    }
    return list;
}
```

פעולות החזרת ערך מספרי

5. פעולה המקבלת רשימה של שלמים, ומחזירה את אורכה

```
public static int Count(Node<int> list)
{
    int count = 0;
    while (list != null)
    {
        count++;
        list = list.GetNext();
    }
    return count;
}
```

5. אותה הפעולה במימוש רקורסיבי

```
public static int CountR(Node<int> list)
{
    if (list == null)
        return 0;
    else
        return 1 + CountR(list.GetNext());
}
```

6. פעולה המקבלת רשימה של שלמים, ומחזירה את סכומה

```
public static int Sum(Node<int> list)
{
    int sum = 0;
    while (list != null)
    {
        sum += list.GetValue();
        list = list.GetNext();
    }
    return sum;
}
```

6. אותה הפעולה במימוש רקורסיבי

```
public static int SumR(Node<int> list)
{
    if (list == null)
        return 0;
    else
        return list.GetValue() + SumR(list.GetNext());
}
```

7. פעולה המקבלת רשימה של שלמים, ומחזירה את ממוצע ערכי החוליות

```
public static double Average(Node<int> list)
{
    return SumR(list) / (double)CountR(list);
}
```

8. פעולה המקבלת רשימה של שלמים, ומחזירה את מכפלת החוליות

```
public static int Product(Node<int> list)
{
    if (list == null)
        return 0;
    int p = 1;      // הכפלה לאפשר....
    while (list != null)
    {
        p *= list.GetValue();
        list = list.GetNext();
    }
    return p;
}
```

8. אותה הפעולה במימוש רקורסיבי

```
public static int ProductR(Node<int> list)
{
    if (list == null)
        return 0;
    if (list.HasNext())
        return list.GetValue() * ProductR(list.GetNext());
    else
        return
            list.GetValue();
}
```

9. פעולה המקבלת רשימה של שלמים, ומחזירה את הערך המקסימלי
הנחה: הרשימה אינה ריקה

```
public static int Max(Node<int> list)
{
    int max = list.GetValue();
    list = list.GetNext();
    while (list != null)
    {
        max = Math.Max(max, list.GetValue());
        list = list.GetNext();
    }
    return max;
}
```

9. אותה הפעולה במימוש רקורסיבי

```
public static int MaxR(Node<int> list)
{
    if(list.HasNext())
        return Math.Max(list.GetValue(), MaxR(list.GetNext()));
    return list.GetValue();
}
```

10. פעולה המקבלת רשימה של שלמים, ומחזירה את הערך המקסימלי
הנחה: הרשימה אינה ריקה

```
public static int Min(Node<int> list)
{
    int min = list.GetValue();
    list = list.GetNext();
    while (list != null)
    {
        min = Math.Min(min, list.GetValue());
        list = list.GetNext();
    }
    return min;
}
```

10. אותה הפעולה במימוש רקורסיבי

```
public static int b(Node<int> list)
{
    if (list.HasNext())
        return Math.Min(list.GetValue(), MinR(list.GetNext()));
    return list.GetValue();
}
```

11. פעולה המקבלת רשימה של שלמים, ומחזירה את סכום כל האיברים הזוגיים

```
public static int SumEvenR(Node<int> list)
{
    if (list == null)
        return 0;
    return (list.GetValue() % 2 == 0 ? list.GetValue() : 0) +
SumEvenR(list.GetNext());
}
```

12. פעולה המקבלת רשימה של שלמים, ומחזירה את סכום כל האיברים הנמצאים המקומות האי-זוגיים. החוליה הראשונה במקום מספר 1

```
public static int SumOddPlaces(Node<int> list)
{
    int sum = 0;
    while (list != null)
    {
        sum += list.GetValue();
        list = list.GetNext();
        if (list != null)
            list = list.GetNext();
    }
    return sum;
}
```

13. פעולה המקבלת רשימה של שלמים ומספר שלם, ומחזירה את מספר הפעמים שהשלם מופיע ברשימה

```
public static int CountX(Node<int> list, int X)
{
    if (list == null)
        return 0;
    return (list.GetValue() == X) ? 1 + CountX(list.GetNext(), X) :
CountX(list.GetNext(), X);
}
```

פעולות המחזירות ערך בוליאני

14. פעולה המקבלת רשימה של שלמים ומחזירה אמת אם הרשימה מסודרת בסדר עולה ממש, שקר אחרת

```
public static bool IsUp(Node<int> list)
{
    if (list == null || !list.HasNext())
        return true;
    while (list.HasNext())
    {
        if (list.GetValue() >= list.GetNext().GetValue())
            return false;
        list = list.GetNext();
    }
    return true;
}
```

15. פעולה המקבלת רשימה של שלמים ומספר שלם ומחזירה אמת אם המספר בשלם נמצא ברשימה שקר אחרת

```
public static bool IsExist(Node<int> list, int x)
{
    if (list == null)
        return false;
    return list.GetValue() == x || IsExist(list.GetNext(), x);
}
```

16. פעולה המקבלת רשימה של שלמים ומחזירה אמת אם הרשימה מהווה סדרה חשבונית, שקר אחרת. סדרה חשבונית ← ההפרש בין כל שני איברים סמוכים זהה

```
public static bool IsArithmetic(Node<int> list)
{
    if (list == null || !list.HasNext())
        return false;
    int x = list.GetNext().GetValue() - list.GetValue();
    list = list.GetNext();
    while (list.HasNext())
    {
        if (list.GetNext().GetValue() - list.GetValue() != x)
            return false;
        list = list.GetNext();
    }
    return true;
}
```

פעולות המחזירות הפנייה למקום ברשימה

17. פעולה המקבלת רשימה של שלמים, ומספר שלם של צעדים, ומחזירה הפנייה לחוליה הנמצאת במקום המבוקש. אם הרשימה ריקה או מספר הצעדים גדול מאורך הרשימה, יוחזר null

```
public static Node<int> Goto(Node<int> list, int step)
{
    while (list != null && step > 1)
    {
        list = list.GetNext();
        step--;
    }
    return list;
}
```

18. פעולה המקבלת רשימה של שלמים ומחזירה הפנייה לחוליה האחרונה ברשימה

```
public static Node<int> GotoLast(Node<int> list)
{
    while (list.HasNext())
        list = list.GetNext();
    return list;
}
```

19. פעולה המקבלת רשימה של שלמים, וערך x, ומחזירה הפנייה למקום ההופעה הראשון של x מתחילת הרשימה. אם לא נמצא כזה, יוחזר null

```
public static Node<int> FirstPosX(Node<int> list, int x)
{
    if (list == null)
        return null;
    if (list.GetValue() == x)
        return list;
    return FirstPosX(list.GetNext(), x);
}
```

20. פעולה המקבלת רשימה של שלמים, ומחזירה הפנייה לערך המינימלי

```
public static Node<int> PosMin(Node<int> list)
{
    Node<int> minPos = list;
    int val, minVal;
    if (list == null)
        return null;
    minVal = list.GetValue();
    list = list.GetNext();
    while (list != null)
    {
        val = list.GetValue();
        if (val < minVal)
        {
            minVal = val;
            minPos = list;
        }
        list = list.GetNext();
    }
    return minPos;
}
```

שינוי רשימה קיימת, בפועל מחזיר רשימה חדשה

21. פעולה המקבלת רשימה של שלמים, מקום ברשימה pos, וערך x ומחזירה רשימה חדשה כאשר הערך x נוסף מקום אחד אחרי pos . אם pos=null , יחזיר רשימה בה x התווסף לפני חוליה ראשונה.

```
public static Node<int> Insert(Node<int> list, Node<int> pos, int x)
{
    if (pos == null)
        return new Node<int>(x, list);
    pos.SetNext(new Node<int>(x, pos.GetNext()));
    return list;
}
```

22. פעולה המקבלת רשימה, מספר צעדים step וערך x, ומחזירה רשימה חדשה בה הערך x נוסף אחרי step צעדים מתחילת הרשימה. אם step=0, x יתווסף לפני החוליה ראשונה. אם step גדול מאורך הרשימה, x יתווסף בסוף הרשימה.

```
public static Node<int> Insert(Node<int> list, int step, int x)
{
    if (step == 0)
        return new Node<int>(x, list);
    Node<int> temp = list;
    while (list != null && list.HasNext() && step > 1)
    {
        list = list.GetNext();
        step--;
    }
    list.SetNext(new Node<int>(x, list.GetNext()));
    return temp;
}
```

23. פעולה המקבלת רשימה ומקום ברשימה prev, ומחזירה רשימה חדשה כאשר החוליה העוקבת ל-prev לא נמצאת ברשימה החדשה. אם prev=null , תחזיר רשימה ללא החוליה הראשונה.

```
public static Node<int> Delete(Node<int> list, Node<int> prev)
{
    if (list == null)
        return null;
    if (prev == null)
    {
        Node<int> tmp = list.GetNext();
        list.SetNext(null);
        return tmp;
    }
    if(prev.HasNext())
        prev.SetNext(prev.GetNext().GetNext());
    return list;
}
```

24. פעולה המקבלת רשימה ומספר צעדים step מתחילת הרשימה ומחזירה רשימה חדשה ללא הערך הנמצא במיקום step .

```
public static Node<int> Delete(Node<int> list, int step)
{
    if (list == null)
        return null;
    Node<int> tmp = list;
    if (step < 2)
    {
        tmp = list.GetNext();
        list.SetNext(null);
        return tmp;
    }

    while (list.HasNext() && step > 2)
    {
        list = list.GetNext();
        step--;
    }
    if (list.HasNext())
    {
        list.SetNext(list.GetNext().GetNext());
    }
    return tmp;
}
```

פעולות על רשימות ממוינות

25. פעולה המקבלת רשימה ממוינת בסדר עולה ומספר שלם x, ומחזירה רשימה חדשה הכוללת את הערך x כך שהיא שומרת על המיון שלה

```
public static Node<int> InsertSort(Node<int> list, int x)
{
    Node<int> nd = new Node<int>(x);
    if (list == null)
        return nd;
    Node<int> tmp = list;
    if (x < list.GetValue())  // add before the first node
    {
        nd.SetNext(list);
        return nd;
    }
    while (list.HasNext() && x > list.GetNext().GetValue())
        list = list.GetNext();
    nd.SetNext(list.GetNext());
    list.SetNext(nd);
    return tmp;
}
```

26. פעולה המקבלת שתי רשימות ממוינות בסדר עולה בעלות ערכים שונים זה מזה ומחזירה רשימה חדשה שהיא מיזוג של שתי הרשימות. כל ערך מופיע רק פעם אחת לכל היותר. הרשימה החדשה מכילה את הערכים משתי הרשימות בסדר עולה.
שים לב: פעולה זו אינה שומרת על הרשימות המקוריות, ומשתמשת בחוליות המקוריות

```
public static Node<int> Merge(Node<int> list1, Node<int> list2)
{
    if (list1 == null)
        return list2;
    if (list2 == null)
        return list1;
    // list1!=null && list2!=null
    Node<int> mrgList;    // refer to merged sortedlist
    // choose reference to mrgList
    if (list1.GetValue() < list2.GetValue())
    {
        mrgList = list1;
        list1 = list1.GetNext();
    }
    else
    {// list1.getValue()>=list2.getValue()
        mrgList = list2;
        list2 = list2.GetNext();
    }
    Node<int> posMrg = mrgList;
    while (list1 != null && list2 != null)
    {
        if (list1.GetValue() < list2.GetValue())
        {
            posMrg.SetNext(list1);
            list1 = list1.GetNext();
        }
```

```
        else
        {// list1.getValue()>=list2.getValue()
            posMrg.SetNext(list2);
            list2 = list2.GetNext();
        }
        posMrg = posMrg.GetNext();
    }// end while
    while (list1 != null)
    {// tail of list1
        posMrg.SetNext(list1);
        list1 = list1.GetNext();
        posMrg = posMrg.GetNext();
    }
    while (list2 != null)
    {// tail of list2
        posMrg.SetNext(list2);
        list2 = list2.GetNext();
        posMrg = posMrg.GetNext();
    }
    return mrgList;
}
```

26. פעולה זהה לפעולת המיזוג הקודמת, אך שומרת על הרשימות המקוריות, ומחזירה רשימה חדשה.

```
public static Node<int> Merge2(Node<int> list1, Node<int> list2)
{
    if (list1 == null)
        return CopyList(list2);
    if (list2 == null)
        return CopyList(list1);

    // list1!=null && list2!=null
    Node<int> mrgList, posMrg;    // refer to merged sortedlist
    // choose reference to mrgList
    if (list1.GetValue() < list2.GetValue())
    {
        mrgList = new Node<int>(list1.GetValue());
        posMrg = mrgList;
        list1 = list1.GetNext();
    }
    else
    {// list1.getValue()>=list2.getValue()
        mrgList = new Node<int>(list2.GetValue());
        posMrg = mrgList;
        list2 = list2.GetNext();
    }

    while (list1 != null && list2 != null)
    {
        if (list1.GetValue() < list2.GetValue())
        {
            posMrg.SetNext(new Node<int>(list1.GetValue()));
            list1 = list1.GetNext();
        }
        else
        {// list1.getValue()>=list2.getValue()
            posMrg.SetNext(new Node<int>(list2.GetValue()));
            list2 = list2.GetNext();
        }
```

```
        posMrg = posMrg.GetNext();
    }// end while
    if (list1 != null)   // tail of list1
        posMrg.SetNext(CopyList(list1));
    if (list2 != null)   // tail of list2
        posMrg.SetNext(CopyList(list2));

    return mrgList;
}
```

26א'. פעולת עזר המקבלת רשימה ומעתיקה אותה לרשימה חדשה.

```
public static Node<int> CopyList(Node<int> list)
{
    if (list == null)
        return null;
    Node<int> newList, lp;
    newList = new Node<int>(list.GetValue());
    lp = newList;
    list = list.GetNext();
    while (list != null)
    {
        lp.SetNext(new Node<int>(list.GetValue()));
        lp = lp.GetNext();
        list = list.GetNext();
    }
    return newList;
}
```

27. פעולה המקבלת שתי רשימות ממוינות בסדר עולה בעלות ערכים שונים זה מזה ומחזירה רשימה חדשה שהיא חיתוך של שתי הרשימות. הרשימה החדשה תכיל את הערכים הנמצאים בשתי הרשימות והיא בסדר עולה.

שים לב: פעולה זו אינה שומרת על הרשימות המקוריות, ומשתמשת בחוליות המקוריות

```
public static Node<int> Cut(Node<int> list1, Node<int> list2)
{
    if (list1 == null || list2 == null)
        return null;

    Node<int> pos, cutList = new Node<int>(0);  // dummy first node;
    pos = cutList;

    while (list1 != null && list2 != null)
    {
        if (list1.GetValue() == list2.GetValue())
        {
            pos.SetNext(list1);
            pos = list1;
            list1 = list1.GetNext();
            list2 = list2.GetNext();
            pos.SetNext(null);
        }
        else if (list1.GetValue() < list2.GetValue())
        {
            list1 = list1.GetNext();
```

```
        }
        else
            list2 = list2.GetNext();
    }

    return cutList.GetNext();   // bypass first dummy node;
}
```

27. פעולה חיתוך השומרת על הרשימות המקוריות, ומחזירה רשימה חדשה

```
public static Node<int> Cut2(Node<int> list1, Node<int> list2)
{
    if (list1 == null || list2 == null)
        return null;

    Node<int> pos, cutList = new Node<int>(0);  // dummy first node;
    pos = cutList;

    while (list1 != null && list2 != null)
    {
        if (list1.GetValue() == list2.GetValue())
        {
            pos.SetNext(new Node<int>(list1.GetValue()));
            pos = pos.GetNext(); ;
            list1 = list1.GetNext();
            list2 = list2.GetNext();
        }
        else if (list1.GetValue() < list2.GetValue())
        {
            list1 = list1.GetNext();
        }
        else
            list2 = list2.GetNext();
    }

    return cutList.GetNext();   // bypass first dummy node;
}
```

פעולות על שתי רשימות

28. פעולה המקבלת שתי רשימות ומחזירה 'אמת' אם כל איברי list2 נמצאים בסדר כלשהו ב-list1, 'שקר'-אחרת.

```
public static bool IsAllExist(Node<int> list1, Node<int> list2)
{
    if (list1 == null || list2 == null)
        return false;

    Node<int> pos1 = list1;
    Node<int> pos2 = list2;
    bool found = false;
    while (pos2 != null)
    {
        while (pos1 != null)
        {
            if (pos1.GetValue() == pos2.GetValue())
                found = true;
            pos1 = pos1.GetNext();
        }
        if (!found)
            return false;
        else
        {
            pos1 = list1;
            pos2 = pos2.GetNext();
            found = false;
        }
    }
    return true;
}
```

29. פעולה המקבלת שתי רשימות list1,list2 ומחזירה 'אמת' אם שתי הרשימות מכילות אותן ערכים ללא חשיבות לסדר, 'שקר'-אחרת.

```
public static bool IsSameValues(Node<int> list1, Node<int> list2)
{
    return IsAllExist(list1, list2) && IsAllExist(list2, list1);
}
```

30. פעולה המקבלת שתי רשימות ומחזירה 'אמת' אם לשתי הרשימות אין ערכים משותפים, 'שקר'-אחרת

```
public static bool IsAllDifferent(Node<int> list1, Node<int> list2)
{
    if (list1 == null || list2 == null)
        return true;
    while (list1 != null && FirstPosX(list2, list1.GetValue()) == null)
        list1 = list1.GetNext();
    if (list1 == null)
        return true;
    else
        return false;
```

```
}
```

31. פעולה המקבלת שתי רשימות list1,list2 ומחזירה 'אמת' אם רשימה list2 מוכלת בשלמותה ברשימה list1 החל מראשית רשימה list1, 'שקר'-אחרת.

```
public static bool IsBeginWith(Node<int> list1, Node<int> list2)
{
    if (list1 == null)
        return false;
    if (list2 == null)
        return true;

    while (list2 != null && list1 != null)
    {
        if (list2.GetValue() != list1.GetValue())
            return false;
        list1 = list1.GetNext();
        list2 = list2.GetNext();
    }
    if (list2 == null)
        return true;
    else
        return false;
}
```

32. פעולה המקבלת שתי רשימות list1,list2 ומחזירה 'אמת' אם רשימה list2 מוכלת בשלמותה ברשימה list1, 'שקר'-אחרת.

```
public static bool IsInclude(Node<int> list1, Node<int> list2)
{
    if (list2 == null)
        return true;
    if (list1 == null && list2 != null)
        return false;
    while (list1 != null)
    {
        if (IsBeginWith(list1, list2))
            return true;
        list1 = list1.GetNext();
    }
    return false;
}
```

פעולות מיון על רשימה

33. פעולה הקולטת מספרים שלמים מהמשתמש, עד לקליטת המספר 99, ובונה ומחזירה רשימה ממוינת של שלמים

```
public static Node<int> CreateSorted()
{
    Node<int> list=null;
    int x;
    Console.WriteLine("Enter integer , 99 to stop");
    x = int.Parse(Console.ReadLine());
    while (x!=99)
    {
        list = InsertSort(list, x);
        Console.WriteLine("Enter integer , 99 to stop");
        x = int.Parse(Console.ReadLine());
    }
    return list;
}
```

34. פעולה המקבלת רשימה של שלמים ממיינת אותה בשיטת מיון הכנסה, ומחזירה רשימה ממוינת

```
public static Node<int> InsertionSort(Node<int> list)
{
    int x;
    Node<int> sortList = null;
    while (list != null)
    {
        x = list.GetValue();
        sortList = InsertSort(sortList, x);
        list = list.GetNext();
    }
    return sortList;
}
```

35. פעולה המקבלת רשימה של שלמים ממיינת אותה בשיטת מיון בחירה, ומחזירה רשימה ממוינת

```
public static void SelectionSort(Node<int> list)
{
    int temp;
    Node<int> pos, minPos;
    pos = list;
    while (pos != null)
    {
        minPos = PosMin(pos);
        if (pos != minPos)
        {
            temp = pos.GetValue();
            pos.SetValue(minPos.GetValue());
            minPos.SetValue(temp);
        }
        pos = pos.GetNext();
    }
}
```