מטלת מנחה (ממ"ן) 14

הקורס: 20441 - מבוא למדעי המחשב ושפת Java

חומר הלימוד למטלה: יחידה 11 **נושא המטלה:** רשימות מקושרות

מספר השאלות: 4 **משקל המטלה:** 3 נקודות

סמסטר: 2024א **מועד אחרון להגשה:** 17.2.2024

במטלה זו נעסוק ברשימות מקושרות של מספרים שלמים.

בשתי השאלות הראשונות נשתמש ברשימה חד-סטרית. לשם כך נכתוב מחלקה בשם IntList המייצגת רשימה מקושרת כזו.

הרשימה תשתמש במחלקה IntNode הבאה, המייצגת איבר ברשימה:

```
public class IntNode {
      private int _value;
      private IntNode _next;

      public IntNode(int val, IntNode n) {
            _value = val;
            _next = n;
      }
      public IntNode(int val) {
            _value = val;
            _next = null;
      }

      public int getValue()  {
             return _value;
      }
      public IntNode getNext()  {
             return _next;
      }

      public void setValue(int v)  {
            _value = v;
      }
      public void setNext(IntNode node)  {
            _next = node;
      }
}
```

אסור להוסיף למחלקה זו תכונות ו/או שיטות פרטיות או ציבוריות!

להלן המחלקה IntList:

```
public class IntList
{
    private IntNode _head;

    public IntList( ) {
        _head = null;
    }

    public void addToEnd(int num) {
    // adds num at the end of the list
        IntNode node = new IntNode(num);
        if (_head == null)
            _head = node;
        else {
            IntNode ptr = _head;
            while (ptr.getNext( ) != null)
                ptr = ptr.getNext( );
            ptr.setNext(node);
        }
    }


    public String toString()
    {
        String s = "";
        IntNode temp = _head;
        while (temp != null)
        {
            s = s+ temp.getValue() + " --> ";
            temp = temp.getNext();
        }
        s+= " null";
        return s;
    }


    //   כאן ייכנסו השיטות שתכתבו בשאלות הבאות
```

הערות חשובות לגבי שאלות 1 ו- 2:

- **השיטה שתכתבו צריכה להיות יעילה ככל הניתן, גם מבחינת סיבוכיות הזמן וגם מבחינת סיבוכיות המקום. תשובה שאינה יעילה מספיק כלומר, שתהיה בסיבוכיות גדולה יותר מזו הנדרשת לפתרון הבעיה תקבל מעט נקודות בלבד.**
- **ניתן להשתמש בשיטות עזר פרטיות ככל הנדרש. בחישוב הסיבוכיות צריך לחשב גם את הזמן והמקום של שיטות העזר.**

- **כתבו (באנגלית בלבד) כחלק מה- API של השאלה מה סיבוכיות הזמן (Time complexity) וסיבוכיות המקום (Space complexity) של השיטה שכתבתם. הסבירו תשובתכם.**

שאלה 1 - 25 נקודות

הוסיפו למחלקה IntList שיטה בוליאנית בשם subListSum המקבלת כפרמטר מספר שלם **חיובי ממש** num. השיטה מחזירה trueאם יש ברשימה תת-רשימה **רצופה** של חוליות שסך הערכים בהן שווה ל-num. אם אין תת-רשימה כזו, השיטה תחזיר false.

לדוגמא,

אם הרשימה היא {3, 6, 2, 7, 1} (ראש הרשימה הוא 3), והפרמטר num=15 השיטה תחזיר true כי סכום התת-רשימה {6, 2, 7} הוא 15.

לעומת זאת אם הפרמטר num = 14, השיטה תחזיר false, כי אין תת-רשימה (רצופה) שסכום ערכיה הוא 14.

הניחו כי ברשימה בשאלה זו יש מספרים חיוביים ממש (ללא אפסים).

חתימת השיטה :

```
public boolean subListSum(int num);
```

שאלה 2 - 25 נקודות

נתונה רשימה מקושרת שאיבריה הם מספרים שלמים (לא רק חיוביים).

על כל חוליה ברשימה ניתן להביט כמחלקת את הרשימה לשני חלקים :

1. כל החוליות מראש הרשימה עד החוליה הזו (כולל) ;
2. כל החוליות מהחוליה שאחרי החוליה הזו ועד לסוף הרשימה.

לכל אחד משני החלקים הללו אפשר לחשב את ממוצע הערכים הנמצאים בחוליות השייכות אליו.

הוסיפו למחלקה IntList שיטה בשם averageNode המחזירה את החוליה המחלקת את הרשימה באופן שבו הערך המוחלט של ההפרש בין ממוצעי שני החלקים יהיה הגדול ביותר האפשרי בהשוואה לכל חלוקה אחרת.

שימו לב: החוליה המוחזרת לא יכולה להיות החוליה האחרונה ברשימה. כלומר, אף אחד מהחלקים שלהם מתחלקת הרשימה לא יכול להיות ריק. אם הרשימה ריקה או בעלת איבר אחד בלבד, השיטה תחזיר null.

דוגמאות:

- **בהינתן הרשימה הבאה: {5, 7, -2, 10}**

החלוקות האפשריות הן:

1. החלק השמאלי הוא {5} והממוצע שלו 5, החלק הימני הוא {7, -2, 10} והממוצע שלו 5 = (7+(-2)+10)/3. ההפרש הוא **0** = 5 – 5
2. החלק השמאלי הוא {5, 7} והממוצע שלו 6 = (5+7)/2, החלק הימני הוא {-2, 10} והממוצע שלו 4 = ((-2)+10)/2. ההפרש הוא **2** = 4 – 6
3. החלק השמאלי הוא {5, 7, -2} והממוצע שלו 3.3333 = (5+7+(-2))/3, החלק הימני הוא {10} והממוצע שלו 10. ההפרש הוא **6.666** = 3.333 – 10

ולכן השיטה תחזיר את החוליה שמכילה את הערך 2-, שמחלקת את הרשימה כך שההפרש בין הממוצעים הוא מקסימלי. שימו לב שהשיטה לא מחזירה את ההפרש אלא את החוליה.

- **בהינתן הרשימה הבאה: {1, 0, 0, 0, 1}**

החלוקות האפשריות הן:

1. החלק השמאלי הוא {1} והממוצע שלו 1, החלק הימני הוא {0, 0, 0, 1} והממוצע שלו 0.25 = (0+0+0+1)/4. ההפרש הוא **0.75** = 0.25 – 1
2. החלק השמאלי הוא {1, 0} והממוצע שלו 0.5= (1+0)/2, החלק הימני הוא {0, 0, 1} והממוצע שלו 0.33 = (0+0+1)/3. ההפרש הוא **0.17** = 0.33 – 0.5
3. החלק השמאלי הוא {1, 0, 0} והממוצע שלו 0.33= (1+0+0)/3, החלק הימני הוא {0, 1} והממוצע שלו 0.5 = (0+1)/2. ההפרש הוא **0.17** = 0.33 – 0.5
4. החלק הימני הוא {1, 0, 0, 0} והממוצע שלו 0.25 = (0+0+0+1)/4, החלק השמאלי הוא {1} והממוצע שלו 1. ההפרש הוא **0.75** = 0.25 – 1

כאן יש שתי חוליות (ה- 1 הראשון משמאל, והאפס הראשון מימין) שבהם ההפרש הוא מקסימלי (0.75) במקרה כזה השיטה תחזיר את החוליה שמכילה את החוליה **האחרונה** בה ההפרש הוא מקסימלי והיא ה- 0 האחרון.

חתימת השיטה :

```
public IntNode averageNode();
```

בשתי השאלות האחרונות נשתמש ברשימה דו-סטרית. לשם כך נכתוב מחלקה בשם IntListTwo המייצגת רשימה מקושרת כזו.

הרשימה תשתמש במחלקה IntNodeTwo הבאה, המייצגת איבר ברשימה:

```
public class IntNodeTwo
{
    private int _num;
    private IntNodeTwo _next, _prev;

    public IntNodeTwo(int n) {
        _num = n;
        _next = null;
        _prev = null;
    }

    public IntNodeTwo(int num, IntNodeTwo n, IntNodeTwo p) {
        _num = num;
        _next = n;
        _prev = p;
    }

    public int getNum()          { return _num;    }
    public IntNodeTwo getNext()  { return _next;   }
    public IntNodeTwo getPrev()  { return _prev;   }
    public void setNum (int n)   { _num = n;       }
    public void setNext (IntNodeTwo node) { _next = node; }
    public void setPrev (IntNodeTwo node) { _prev = node; }
}
```

אסור להוסיף למחלקה זו תכונות ו/או שיטות פרטיות או ציבוריות!

להלן המחלקה IntListTwo:

```
public class IntListTwo
{
    IntNodeTwo _head, _tail;
    // שימו לב שלא להגדיר את התכונות האלו כפרטיות!

    public IntListTwo()
    {
        _head = null;
        _tail = null;
    }


    //   כאן ייכנסו השיטות שתכתבו בשאלות הבאות
```

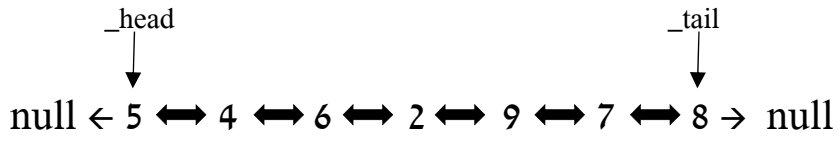
שאלה 3 - 25 נקודות

1. הוסיפו למחלקה IntListTwo שיטה המקבלת כפרמטר מספר שלם ומוסיפה אותו לסוף הרשימה הדו-סטרית.

חתימת השיטה: `public void addToEnd (int num)`

2. הוסיפו למחלקה IntListTwo את השיטה toString המחזירה את הרשימה כמחרוזת תווים מתחילת הרשימה לסופה, לפי הדוגמא הבאה **בדיוק**:

אם נתונה הרשימה list הבאה:



המחרוזת שתוחזר תהיה `{5, 4, 6, 2, 9, 7, 8}`

חתימת השיטה: `public String toString()`

שימו לב שהמחרוזת המוחזרת תהיה בדיוק כמו זו שבדוגמא. יש רווח אחד לאחר כל פסיק. ללא סימנים נוספים!

3. הוסיפו למחלקה IntListTwo את השיטה **הרקורסיבית** toStringReverse המחזירה את הרשימה כמחרוזת תווים מסוף הרשימה לתחילתה, לפי הדוגמא הבאה **בדיוק**:

אם הרשימה list היא כמו לעיל, המחרוזת שתוחזר תהיה:

`{8, 7, 9, 2, 6, 4, 5}`. גם כאן ללא רווחים מיותרים וללא סימנים נוספים.

השיטה toStringReverse **חייבת להיות רקורסיבית וללא שימוש בלולאות כלל!**

חתימת השיטה: `public String toStringReverse()`

שאלה 4 - 25 נקודות

נתונה רשימה מקושרת דו-סטרית שבכל אחת מחוליותיה יש מספר שלם **חיובי**.
נגדיר **מסלול חוקי** ברשימה כסדרה של חוליות ברשימה, המתחילה בראש הרשימה ומתקדמת ברשימה מספר צעדים **ימינה או שמאלה** לפי הערך שבחוליה. שימו לב שאי אפשר להתקדם מעבר לגבולות הרשימה. המסלול צריך להסתיים בחוליה האחרונה ברשימה.
כתבו שיטה **רקורסיבית** בוליאנית המקבלת רשימה כמתואר לעיל ומחזירה true אם ישנו מסלול חוקי ברשימה, ו-false אחרת.

- עבור הרשימה $\{2, 4, 1, 6, 4, 2, 4, 3, 5\}$:

התשובה שתוחזר תהיה true שכן ישנו מסלול שמתחיל באיבר שבראש הרשימה (ערכו 2), הולך שני צעדים ימינה לאיבר שערכו 1, משם צעד אחד שמאלה לאיבר שערכו 4, משם ארבעה צעדים ימינה לאיבר שערכו 2, משם שוב שני צעדים ימינה לאיבר שערכו 3, משם שלושה צעדים שמאלה לאיבר שערכו 4 ומשם ארבעה צעדים ימינה לאיבר שערכו 5 שהוא האחרון ברשימה (הזנב).

- עבור הרשימה $\{1, 4, 3, 1, 2, 4, 3\}$:

התשובה שתוחזר תהיה false שכן אין אף מסלול שמתחיל בראש הרשימה ומגיע לזנבה לפי הקפיצות ימינה או שמאלה. אם נתחיל בראש הרשימה, נזוז צעד אחד ימינה לאיבר שערכו 4, משם חייבים לזוז ימינה ארבעה צעדים, כי שמאלה אי אפשר בגלל גבולות הרשימה, וכך מגיעים לאיבר שערכו 4 (שוב) משם חייבים לזוז שמאלה ארבעה צעדים, כי ימינה אי אפשר, (שימו לב שמאיבר זה אין אפשרות להתקדם 4 צעדים ימינה, כיוון שמסתיימת הרשימה) ולכן מגיעים שוב לאיבר השני (שערכו 4) כך שזהו תהליך אינסופי שלא מגיע לסוף הרשימה לעולם.
שימו לב להימנע מרקורסיות אינסופיות כאלו.

חתימת השיטה היא :

```
public boolean isWay()
```

השיטה צריכה להיות רקורסיבית ללא שימוש בלולאות כלל. כך גם כל שיטות העזר שתכתבו (אם תכתבו) לא יכולות להכיל לולאות.
אפשר לשנות את הרשימה המקורית.
אפשר להשתמש בהעמסת-יתר (overloading).
אפשר להוסיף למחלקה שיטות פרטיות בלבד, ובלבד שגם הן לא יכילו לולאות.

הגשה:

1. הגשת הממ״ן נעשית בצורה אלקטרונית בלבד, דרך מערכת שליחת המטלות.
2. הקפידו ששמות המחלקות והשיטות יהיו בדיוק כפי שמוגדר בממ״ן. **אחרת המחלקה לא תעבור קומפילציה והציון יהיה 0.**
3. עליכם להריץ את הטסטרים שנמצאים באתר הקורס על המחלקות שכתבתם. שימו לב שהטסטרים לא מכסים את כל האפשרויות, ובפרט לא את מקרי הקצה. הם רק בודקים את השמות של השיטות במחלקות. מאד מומלץ להוסיף להם בדיקות. שימו לב שאם הטסטרים לא יעברו קומפילציה מול המחלקות שכתבתם, הציון על המטלה יהיה אפס. אם יש שיטה שאתם מעוניינים לדלג עליה, עלכם לרשום את חתימת השיטה ולהחזיר ערך סתמי על מנת שהטסטרים יעברו קומפילציה.
4. את התשובות לשאלות 1 - 2 יש להגיש בקובץ Java ששמו IntList
5. את התשובות לשאלות 3 – 4 יש להגיש בקובץ Java ששמו IntListTwo
6. שימו לב ששני הקבצים שאתם שולחים חייבים להיות העתק מדויק (copy and paste) של הקבצים מהממ״ן, רק עם התוספות שלכם.
7. שימו לב שהתכונות של ראש וזנב הרשימה בקובץ IntListTwo לא הוגדרו ב-private. זאת לא טעות אלא נועד להקל על בדיקת הממ״ן שלכם. כלומר, באופן יוצא דופן (ובניגוד למה שעליכם לעשות בכל הקשר אחר) בראש המחלקה שלכם (שתועתק מהממ״ן, כמובן) צריך להופיע בדיוק כך :

```
;IntNodeTwo _head, _tail
```

ולא

```
;private IntNodeTwo _head, _tail
```

8. **אין להגיש את קובצי ה- API שכתבתם, וגם לא את הקבצים IntNode, IntNode|Two**.
9. **ארזו את שני הקבצים בקובץ zip יחיד ושלחו אותו בלבד.**

ב ה צ ל ח ה