משרד העבודה
MINISTRY OF LABOR

מה"ט
המכון הממשלתי להכשרה טכנולוגית

מספר ת"ז _____
מספר מחברת _____

מועד הבחינה: קיץ תשפ"ד – 2024 – מועד ב'
מספר השאלון: 97105
נספח ממשקים לבחינה JAVA
נספח ממשקים לבחינה C#
מילון עזר

מבני נתונים ותכנות מונחה עצמים

הנדסאים וטכנאים – הנדסת תוכנה

הנחיות לבחינה

א. משך הבחינה: ארבע שעות וחצי.

ב. מבנה השאלון ומפתח ההערכה: **בשאלון זה שני מבחנים, <u>עליכם לענות על מבחן אחד בלבד בהתאם למוסד הלימודים</u>:**

מבחן ב- Java (עמוד 2)

מבחן ב- C# (עמוד 17)

בכל מבחן עשר שאלות.

חלק א' – 48 נקודות

שאלות 1-5: יש לענות על **שלוש** שאלות בלבד. ערך כל שאלה 16 נקודות.

חלק ב' – 36 נקודות

שאלות 6-10: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 18 נקודות.

חלק ג' – 16 נקודות

שאלות 11-12: יש לענות על שאלה **אחת** בלבד. ערך השאלה 16 נקודות.

בסך הכול: 100 נקודות.

ג. חומר עזר מותר לשימוש:

1. מחשבון (אין להשתמש במחשב כף יד או במחשבון עם תקשורת חיצונית).
2. קלסר **<u>אחד בלבד</u>** עם **<u>חומר ההרצאות</u>**. אין להוציא דפים מהקלסר.

אין לצרף ספרים או חוברות עם פתרונות.

ד. הוראות כלליות:

1. יש לקרוא בעיון את ההנחיות בדף השער ואת כל שאלות הבחינה, ולוודא שהן מובנות.
2. את התשובות יש לכתוב בצורה מסודרת, בכתב יד ברור ונקי (גם בכך תלויה הערכת הבחינה).
3. יש להשאיר את העמוד הראשון במחברת הבחינה ריק. בסיום המבחן יש לרשום בעמוד זה את מספרי התשובות לבדיקה. התשובות ייבדקו לפי סדר כתיבתן בעמוד זה. לא ייבדקו תשובות עודפות.
4. יש לכתוב את התשובות במחברת הבחינה **בעט בלבד**, בכתב יד ברור.
5. יש להתחיל כל תשובה בעמוד חדש ולציין את מספר השאלה ואת הסעיף. אין צורך להעתיק את השאלה עצמה.
6. טיוטה יש לכתוב במחברת הבחינה בלבד. יש לרשום את המילה "טיוטה" בראש העמוד ולהעביר עליו קו כדי שלא ייבדק.
7. יש להציג פתרון מלא ומנומק, כולל חישובים לפי הצורך. הצגת תשובה סופית ללא שלבי הפתרון לא תזכה בניקוד.
8. יש להסביר בפירוט כל תוכנית שנכתבה, **תוכנית ללא הסבר מפורט לא תזכה בניקוד**.
9. אם לדעתכם חסר בשאלה נתון, יש לציין זאת ולהוסיף נתון מתאים שיאפשר לכם להמשיך בפתרון השאלה, נמקו את בחירתכם.

חל איסור מוחלט להוציא שאלון או מחברת בחינה מחדרה בחינה!

בהצלחה!

בשאלון זה 21 עמודי בחינה ו- 4 עמודי נספחים.

מבחן ב- JAVA

חלק א'

ענו על 3 מבין השאלות 1-5 (ערך כל שאלה – 16 נקודות).

שאלה 1

א. **(12 נק')** מחסנית של מספרים שלמים נקראת "מחסנית שוויון" אם היא עונה על שלושת התנאים הבאים:

- במחסנית יש רק ערכים החיוביים.
- מספר האיברים הזוגיים שווה למספר האיברים האי-זוגיים.
- סכום האיברים הזוגיים שווה לסכום האיברים האי-זוגיים.

כתבו פעולה המקבלת מחסנית של מספרים שלמים ובודקת אם היא "מחסנית שוויון".
יש לשמור על תוכן של המחסנית אחרי ביצוע הפעולה. כותרת הפעולה:

```
public static boolean isParityStack(Stack<Integer> st)
```

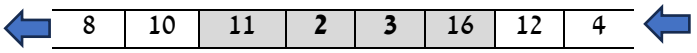
ב. **(4 נק')** מהי סיבוכיות הפעולה שכתבתם בסעיף א'? **הסבירו את תשובתכם.**

שאלה 2

א. **(12 נק')** נתון התור q של מספרים שלמים. כל איבר בתור מופיע פעם אחת (אין חזרות). המרחק בין שני איברים בתור הוא כמות האיברים הנמצאים ביניהם.
כתבו פעולה המקבלת תור q ושני מספרים נוספים x ו- y. הפעולה **תחשב ותחזיר** את המרחק בין שני איברי התור שערכם x ו- y.

לדוגמה:

עבור התור q הבא ופרמטרים x=10, y=12 הפעולה תחזיר 4 כי בין הערכים 10 ו-12 יש בתור ארבעה איברים:



עבור התור q ופרמטרים x=12, y=10 הפעולה גם תחזיר 4 כי בין הערכים 10 ו- 12 ישנם בתור ארבעה איברים.
אם בתור אין איבר שערכו x או אין איבר שערכו y, הפעולה תחזיר ערך 1-.
כותרת הפעולה:

```
public static int distance(Queue<Integer> q, int x, int y)
```

ב. **(4 נק')** מהי סיבוכיות הפעולה שכתבתם בסעיף א'? **הסבירו את תשובתכם.**

שאלה 3

א. **(12 נק')** שרשרת נקראת "שרשרת חזרות" אם כל ערך שבה מופיע **בדיוק פעמיים**. כתבו פעולה המקבלת הפניה לחוליה ראשונה של שרשרת תווים. הפעולה תבדוק אם השרשרת היא "שרשרת חזרות". אם כן – הפעולה תחזיר ערך true, ולא – הפעולה תחזיר ערך false.
כותרת הפעולה:

```
public static boolean repeatList(Node<Character> chain)
```

ב. **(4 נק')** מהי סיבוכיות הפעולה שכתבתם בסעיף א'? **הסבירו את תשובתכם.**

שאלה 4

ברשות הדואר החליטו למחשב את המערך לקבלת דברי הדואר. לצורך כך הוגדרו שלוש מחלקות שונות: `MailItem`, `Letter` ו- `Package`.

המחלקה `Letter` מייצגת מכתב. בעבור כל מכתב יש לשמור את הנתונים הבאים:

- שם שולח המכתב, מטיפוס מחרוזת, `String`.
- מען המכתב (כתובת), מטיפוס מחרוזת, `String`.
- דואר רשום מטיפוס בוליאני, `boolean`: `true` – עבור דואר רשום, `false` – עבור דואר רגיל.
- גודל המכתב, טיפוס תווי, `char` ('S' – עבור מכתב קטן, 'B' – עבור מכתב בגודל סטנדרטי, 'L' – עבור מכתב גדול).
- מחיר בסיסי למשלוח מכתב בגודל סטנדרטי, מטיפוס מספר ממשי, `double`

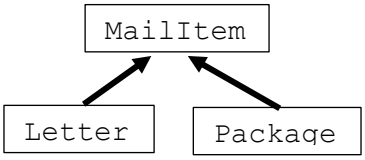
מחיר משלוח מכתב תלוי בגודלו ובסוגו: בעבור מכתב בדואר רשום יש להוסיף 10 ₪ , עבור מכתב קטן יש הנחה של שני שקלים, עבור מכתב גדול יש תוספת של 3 ₪

המחלקה `Package` מייצגת חבילה. בעבור כל מכתב יש לשמור את הנתונים הבאים:

- שם שולח החבילה, מטיפוס מחרוזת, `String`.
- מען המכתב (כתובת), מטיפוס מחרוזת, `String`.
- משקל החבילה בקילוגרמים מטיפוס מספר שלם, `int`.
- האם שביר?: `true` – אם תוכן החבילה שביר, `false` – אם לא שביר.
- מחיר בסיסי למשלוח חבילה מטיפוס מספר ממשי, `double`.

מחיר משלוח חבילה תלוי בגודל החבילה וסוגה: בעבור משקל פחות מ- 5 קילוגרמים יש לשלם מחיר בסיסי. מעל 5 קילוגרמים יש לשלם תוספת של 3 ₪ <u>לכל קילוגרם נוסף</u>. אם התוכן שביר יש להוסיף עוד 15 ₪.

המחלקה `MailItem` היא מחלקת האב למחלקות `Letter` ו-`Package` (המחלקות `Letter` ו- `Package` יורשות מהמחלקה `MailItem`) בהתאם לעקרונות תכנות מונחה עצמים:



א. **(6 נק׳)** כתבו כותרות ותכונות של כל אחת מהמחלקות `MailItem`, `Letter` ו- `Package`.

ב. **(4 נק׳)** כתבו פעולה `public double getRealPrice()`.
הפעולה תחשב ותחזיר מחיר סופי של משלוח. יש לציין באיזה מחלקה/מחלקות צריך להוסיף את הפעולה בהתאם לעקרונות תכנות מונחה עצמים.

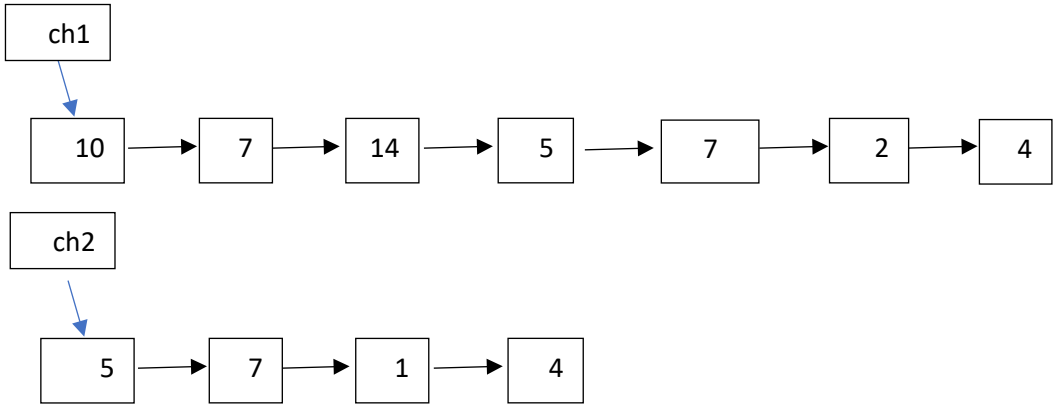
ג. **(6 נק׳)** כתבו פעולה
```
public static String bigSender(MailItem[] args)
```
הפעולה מקבלת מערך של משלוחים ומחזירה את שם השולח של החבילה היקרה ביותר (חבילה שמחיר המשלוח הוא הגבוה ביותר). אם אין אף חבילה במערך, הפעולה תחזיר ערך `null`. אפשר להניח שיש רק משלוח אחד כזה. אפשר להניח שקיימות פעולות `set/get` בכל מחלקה.

שאלה 5

א. **(1 נק')** נתונה הפעולה what הבאה, מה מבצעת הפעולה?

```
public static int what(Node<Integer> ch){
        int c = 0;
        while(ch != null)
        {
            c++;
            ch = ch.getNext();
        }
        return c;
    }
```

ב. **(4 נק')** נתונה הפעולה why המשתמשת בפעולה what, מה תחזיר הפעולה why(ch1, ch2)?
עבור שתי שרשרות החוליות הבאות: **יש להראות מעקב אחרי ביצוע הפעולה why**.
אין צורך במעקב אחרי **what**.




```
public static boolean why(Node<Integer> ch1, Node<Integer> ch2)
{
        int len1 = what(ch1);
        int len2 = what(ch2);
        for(int i = 0; i < len1-len2; i++)
        {
            ch1 = ch1.getNext();
        }
        while(ch1!=null)
        {
            if(ch1.getValue() != ch2.getValue())
                return false;
            ch1 = ch1.getNext();
            ch2 = ch2.getNext();
        }
        return true;
 }
```

ג. **(4 נק')** תנו דוגמה של שרשרת ch3 באורך של **ארבע** חוליות, באופן שבו זימון הפעולה why(ch1, ch3) יחזיר ערך שונה מזה שהוחזר בסעיף ב'.

ד. **(4 נק')** תנו דוגמה של שרשרת ch4 באורך של **ארבע** חוליות, באופן שבו זימון הפעולה why(ch4, ch1) יחזיר ערך true. אם אי אפשר ליצור שרשרת כזאת, הסבירו למה.

ה. **(3 נק')** מה מבצעות הפעולות what(ch) ו- why(ch1,ch2) באופן כללי?

חלק ב'

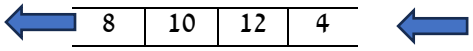
ענו על 2 מבין השאלות 6-10 (ערך כל שאלה – 18 נקודות).

שאלה 6

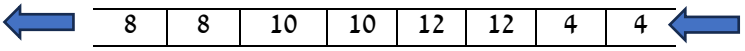
א. **(7 נק')**

כתבו פעולה `void one(Queue<Integer> q, int k)` המקבלת תור של מספרים שלמים `q` ומספר שלם וחיובי `k`. הפעולה "תכפיל" את התור באופן הבא:

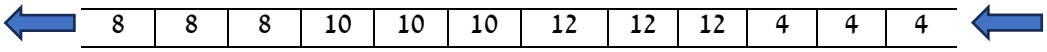
אם לפני זימון הפעולה התור `q` היה כך:



אחרי זימון הפעולה `one(q, 2)` התור `q` יהיה כך:



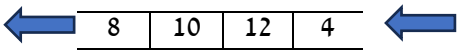
אחרי זימון הפעולה `one(q, 3)` התור `q` יהיה כך:



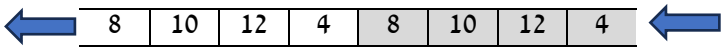
ב. **(7 נק')**

כתבו פעולה `void two(Queue<Integer> q, int k)` המקבלת תור של מספרים שלמים `q` ומספר שלם וחיובי `k`. הפעולה "תכפיל" את תור בצורה הבאה:

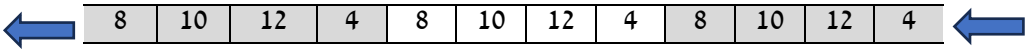
אם לפני זימון הפעולה התור `q` היה:



אז אחרי ביצוע זימון הפעולה `two(q, 2)` התור `q` יהיה:



אז אחרי ביצוע זימון הפעולה `two(q, 3)` התור `q` יהיה:



ג. **(4 נק')** מהן הסיבוכיות של הפעולות שכתבתם בסעיפים א'-ב'? **הסבירו את תשובתכם.**

שאלה 7

נתונה מחלקה המתארת אבן דומינו (Domino):

```
public class Domino
{
 private int left; // ערך מספרי צד שמאלי בין 0 ל-6 (כולל)
 private int right; // ערך מספרי צד ימני בין 0 ל-6 (כולל)

 public Domino(int left, int right)
 {
  this.left = left;
  this.right = right;
 }
}
```

במחלקה הוגדרו פעולות set/get

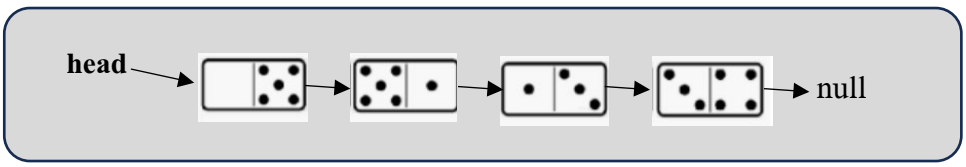
א. **(4 נק')** כתבו במחלקה Domino את הפעולה rotate(). הפעולה מסובבת את האבן הנוכחית כלומר: מחליפה בין הערכים left ו- right.

נתונה המחלקה Deck המתארת רשימת אבני דומינו:

```
public class Deck
{
  private Node<Domino> head;
  public Deck()
  {
    this.head = null;
  }
}
```

רשימת אבני דומינו נקראת "מסודרת היטב" אם ערך של מספרי צד ימני של כל אבן (פרט לאבן האחרונה ברשימה) זהה לערך של מספרי צד שמאלי של האבן הבאה.

לדוגמה: הרשימה הבאה מוגדרת "מסודרת היטב".



רשימה ריקה או רשימה שיש בה רק אבן אחת, היא רשימה "מסודרת היטב".

ב. **(7 נק')** כתבו במחלקה Deck פעולה בוודקת שרשימה היא "מסודרת היטב". אם כן – הפעולה תחזיר ערך true, ולא – תחזיר ערך false.

ג. **(7 נק')** כתבו במחלקה Deck פעולה המקבלת אבן דומינו ובודקת אם אפשר להוסיף אותה לרשימה "מסודרת היטב" כך שהיא תישאר להיות "מסודרת היטב". אם כן – הפעולה תוסיף את האבן ותחזיר ערך true, ולא – תחזיר ערך false.

לדוגמה:
אפשר להוסיף לרשימה שבדוגמה בסעף א' את האבנים הבאות (0,4), (1,1), (4,6).
אי אפשר להוסיף את האבנים הבאות: (3,2), (5,6), (2,2).

<u>שאלה 8</u>

הגדרה: עץ **"מסודר"** הוא עץ בינארי של מספרים שלמים העונה על הכללים הבאים:

- לכל צומת פרט לעלים יש ערך זוגי.
- הערכים של כל העלים הם מספרים אי-זוגיים.
- הערך של צומת הבן גדול מהערך של צומת האב.

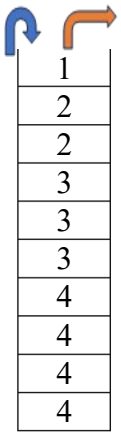
א. **(4 נק')** ציירו עץ "מסודר" הכולל לפחות שישה צמתים.

ב. **(14 נק')** כתבו פעולה המקבלת הפניה לשורש של עץ בינרי של מספרים שלמים. הפעולה תבדוק אם העץ הוא "עץ מסודר". אם כן – הפעולה תחזיר ערך `true`,
ולא – הפעולה תחזיר ערך `false`.

<u>שאלה 9</u>

מחסנית "פירמידה n" היא מחסנית של מספרים שלמים וחיוביים שהמספר הראשון (מספר שנימצא בראש המחסנית) הוא 1 וכל מספר נוסף מופיע בו ברצף, מספר פעמים השווה לערכו, עד N. כלומר, 1 יופיע פעם אחת, לאחריו 2 יופיע שתי פעמיים, מספר 3 מופיע שלוש פעמים וכו'.

לדוגמה: המחסנית "פירמידה 4":



א. **(7 נק')** כתבו פעולה המקבלת מספר שלם וחיובי **n** ומחזירה מחסנית "פירמידה n".

ב. **(7 נק')** כתבו פעולה המקבלת מחסנית של מספרים שלמים. הפעולה תבדוק אם המחסנית היא "פירמידה n". אם כן – הפעולה תחזיר ערך `true`,
ולא – הפעולה תחזיר ערך `false`.

ג. **(4 נק')** מהן סיבוכיות הפעולות שכתבתם בסעיפים א' ו- ב'? **הסבירו את תשובתכם**.

שאלה 10

נתונות המחלקות A, B, C הבאות:

```
public class A
{
    private static int countA = 0;
    protected int x;
    public A()
     {
          countA++;
          this.x = 0;
    }
    public A(int x)
     {
          countA++;
          this.x = x;
    }
    public String toString()
     {
          return "A:"+this.x;
     }
    public int doIt(int z)
     {
          return this.x * z;
     }
    public static int getCount(){ return countA;  }
}// end of class A

public class B extends A
{
    protected int y;
    private static int countB = 0;
    public B(int y)
    {
          this.y = y;
          countB++;
    }
    public B(int x, int y)
    {
          super(x);
          countB++;
          this.y = y;
    }
    public String toString()
    {
        return "B:"+super.toString()+":"+this.y;
    }
    public static int getCount(){return countB;}
    public int doIt(int z)
    {
          return super.doIt(z) + this.y;
     }
```

```
} //end of class B
public class C extends B
{
    private A myA;
    private static int countC = 0;
    public C(int x, int y)
    {
        super(x, y);
        this.myA = new A(x+y);
        countC++;
    }
    public static int getCount(){return countC;}
    public String toString()
    {
        return "C:"+super.toString()+" ["+this.myA+"]";
    }
}// end of class C
```

א. **(1 נק׳)** ציירו תרשים UML שמייצג את הקשרים בין המחלקות A, B, C.
יש לסמן ירושה באמצעות חץ ⟵ והכלה באמצעות הסימן ◆⟵

ב. **(5 נק׳)** נתון קטע הקוד הבא, בפעולה הראשית במחלקה אחרת:

```
A[] arr = new A[5];
arr[0] = new A(13);
arr[1] = new B(5, 10);
arr[2] = new C(2, 7);
arr[3] = new C(3, 3);
arr[4] = new B(5);
```

עקבו אחרי ביצוע קטע הקוד. יש לצייר את כל העצמים שנוצרו ולציין את ערכי התכונות שלהם.

ג. **(3 נק׳)** לקוד של סעיף ב׳ התווסף הקוד הבא, רשמו מה יהיה הפלט בהתאם:

```
System.out.println("A objects: " + A.getCount());
System.out.println("B objects: " + B.getCount());
System.out.println("C objects: " + C.getCount());
```

ד. **(5 נק׳)** לקוד של סעיף ב׳ התווסף הקוד הבא, רשמו מה יהיה הפלט:

```
for (int k=0; k <arr.length; k++)
 System.out.println(arr[k]);
```

ה. **(4 נק׳)** לקוד של סעיף ב׳ התווסף הקוד הבא, רשמו מה יהיה הפלט:

```
for (int k=0; k <arr.length; k++)
 if(arr[k] instanceof B)
     System.out.println(arr[k].doIt(k+1));
```

חלק ג׳

ענו על 1 מבין השאלות 11-12 (ערך שאלה – 16 נקודות).

שאלה 11

החברת **Helpl4U** נותנת שירותי תמיכה טלפונית. מוקד השירות כולל 15 עמדות תמיכה הממוספרות 0-14. שירותי התמיכה מתנהלים באופן הבא:

- כל מתקשר משאיר את שמו ואת מספר הטלפון שלו לרכז מערכת.
- הרכז מצרף את פרטי המתקשר לאוסף הממתינים בעמדת התמיכה הכי פחות עמוסה.
- כאשר עמדת התמיכה מסיימת טיפול במתקשר, פרטיו נמחקים מאוסף הממתינים של העמדה.

מערכת לניהול שירות כוללת כמה מחלקות.

המחלקה **Caller** (מתקשר). למחלקה הוגדרו פעולה בונה ופעולות `getName()` ו-`getPhone()`:

`Caller (String name, String phone)`	הפעולה בונה מתקשר חדש ששמו `name` ומספר הטלפון שלו `phone`

המחלקה **Service** מייצגת עמדת שירות. לפניכם ממשק חלקי של המחלקה `Service`:

void `addCaller (Caller c)`	מוסיפה מתקשר `c` לאוסף הממתינים בעמדה
Caller `callback()`	הפעולה מחזירה את המתקשר הראשון מבין הממתינים בעמדה, ומוחקת את נתוניו מאוסף הממתינים. **הנחה**: יש ממתינים בעמדה
int `getNumOfCallers()`	הפעולה מחזירה את מספר ממתינים בעמדה

א. **(3 נק׳)** כתבו את כותרת המחלקה `Service` ואת התכונות שלה

לייצוג אוסף, אפשר להשתמש במחלקות Stack, Queue, Node. אפשר להוסיף תכונות נוספות. חובה לתעד את התכונות.

ב. **(3 נק׳)** כתבו את הפעולה `callback()`

ניהול מוקד השירות נעשה בעזרת המחלקה **Support**. לפניכם ממשק חלקי של המחלקה `Support`:

void `startSupport (String name, String phone)`	הפעולה מקבלת פרטים של מתקשר ומוסיפה אותו לסוף אוסף הממתינים של העמדה שמספר הממתינים בה הוא הנמוך ביותר. אם יש כמה עמדות כאלה, המתקשר יתווסף לראשונה מבניהן.

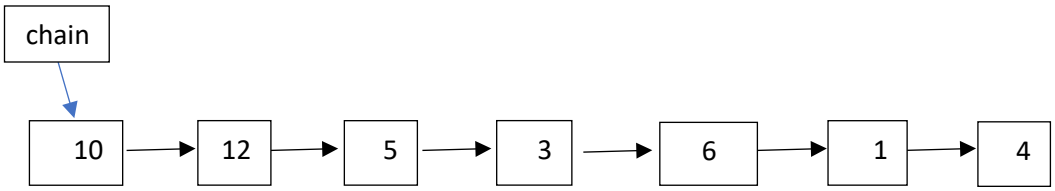
ג. **(3 נק׳)** כתבו את כותרת המחלקה `Support` ואת התכונות שלה.

ד. **(3 נק׳)** כתבו את הפעולה `startSupport()`.

ה. **(4 נק׳)** הנהלת החברה החליטה לציין בסוף כל יום עבודה את פקיד/ת השרות שטיפל/ה בהכי הרבה לקוחות. ציינו שינוים הנדרשים במחלקה/מחלקות וכתבו במחלקה `Support` פעולה `theBest()` המחזירה את מספר העמדה של הפקיד הטוב ביותר (אפשר להניח שיש רק פקיד אחד).

<u>שאלה 12</u>

נתונה השרשרת הבאה:



א. **(6 נק׳)** עקבו אחרי זימון הפעולה הרקורסיבית `one(chain)` הבאה וכתבו איך תיראה השרשרת אחרי ביצוע הפעולה? כמו כן, כתבו מה מבצעת הפעולה `one` באופן כללי?

יש להראות מעקב אחרי ביצוע הפעולה!

```
public static void one(Node<Integer> n){
    if(n.hasNext())
    {
        n.setValue(n.getNext().getValue());
        one(n.getNext());
    }
    else
    {
        n.setValue(0);
    }
}
```

ב. **(6 נק׳)** נתונות הפעולה הרקורסיבית `two` הבאה, עקבו אחרי זימון הפעולה `two(chain,10)` ורשמו איך תיראה השרשרת אחרי ביצוע הפעולה? רשמו כמו כן מה מבצעת הפעולה באופן כללי?

יש להראות מעקב אחרי ביצוע הפעולה!

```
public static void two(Node<Integer> n, int num){
    if(n.hasNext())
    {
        two(n.getNext(), num);
    }
    else
    {
        n.setValue(num);
    }
}
```

ג. **(4 נק׳)** נתונה הפעולה `three` הבאה, מה יהיה תוכן השרשרת `chain` אחרי זימון הפעולה `three(chain)`? כתבו מה מבצעת הפעולה באופן כללי?

```
public static void three(Node<Integer>n){
    int x = n.getValue();
    one(n);
    two(n,x);
}
```

מבחן ב- C#
חלק א'
ענו על 3 מבין השאלות 1-5 (ערך כל שאלה – 16 נקודות).

שאלה 1

א. **(12 נק')** מחסנית של מספרים שלמים נקראת "מחסנית שוויון" אם היא עונה על שלושת התנאים הבאים:

- במחסנית יש רק ערכים החיוביים
- מספר האיברים הזוגיים שווה למספר האיברים האי-זוגיים
- סכום האיברים הזוגיים שווה לסכום האיברים האי-זוגיים

כתבו פעולה המקבלת מחסנית של מספרים שלמים ובודקת אם היא "מחסנית שוויון".
יש לשמור על תוכן של המחסנית אחרי ביצוע הפעולה. כותרת הפעולה:

```
public static bool IsParityStack(Stack<int> st)
```

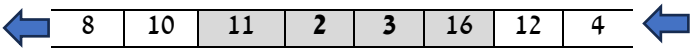
ב. **(4 נק')** מהי סיבוכיות הפעולה שכתבתם בסעיף א'? **הסבירו את תשובתכם.**

שאלה 2

א. **(12 נק')** נתון התור q של מספרים שלמים. כל איבר בתור מופיע פעם אחת (אין חזרות). המרחק בין שני איברים בתור הוא כמות האיברים הנמצאים ביניהם.
כתבו פעולה המקבלת תור q ושני מספרים נוספים x ו-y. הפעולה **תחשב ותחזיר** את המרחק בין שני איברי התור שערכם x ו-y.

לדוגמה:

עבור התור q הבא ופרמטרים x=10, y=12 הפעולה תחזיר 4 כי בין הערכים 10 ו- 12 יש בתור ארבעה איברים:



עבור התור q ופרמטרים x=12, y=10 הפעולה גם תחזיר 4 כי בין הערכים 10 ו-12 ישנם בתור ארבעה איברים.
אם בתור אין איבר שערכו x או אין איבר שערכו y, הפעולה תחזיר ערך 1-.
כותרת הפעולה:

```
public static int Distance(Queue<int> q, int x, int y)
```

ב. **(4 נק')** מהי סיבוכיות הפעולה שכתבתם בסעיף א'? **הסבירו את תשובתכם.**

שאלה 3

א. **(12 נק')** שרשרת נקראת "שרשרת חזרות" אם כל ערך שבה מופיע **בדיוק פעמיים**. כתבו פעולה המקבלת הפניה לחוליה ראשונה של שרשרת תווים. הפעולה תבדוק אם השרשרת היא "שרשרת חזרות". אם כן – הפעולה תחזיר ערך true, ולא – הפעולה תחזיר ערך false.
כותרת הפעולה:

```
public static bool RepeatList(Node<char> chain)
```

ב. **(4 נק')** מהי סיבוכיות הפעולה שכתבתם בסעיף א'? **הסבירו את תשובתכם.**

<u>שאלה 4</u>

ברשות הדואר החליטו למחשב את המערך לקבלת דברי הדואר. לצורך כך הוגדרו שלוש מחלקות שונות: `MailItem`, `Letter` ו- `Package`.

המחלקה `Letter` מייצגת מכתב. בעבור כל מכתב יש לשמור את הנתונים הבאים:

- שם שולח המכתב, מטיפוס מחרוזת, `string`
- מען המכתב (כתובת), מטיפוס מחרוזת, `string`
- דואר רשום מטיפוס בוליאני, `bool`: `true` – עבור דואר רשום, `false` – עבור דואר רגיל.
- גודל המכתב, טיפוס תווי, `char` ('S' – עבור מכתב קטן, 'B' – עבור מכתב בגודל סטנדרטי, 'L' – עבור מכתב גדול)
- מחיר בסיסי למשלוח מכתב בגודל סטנדרטי, מטיפוס מספר ממשי, `double`

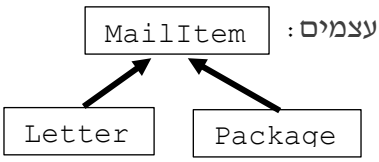
מחיר משלוח מכתב תלוי בגודלו ובסוגו: בעבור מכתב בדואר רשום יש להוסיף 10 ₪ , עבור מכתב קטן יש הנחה של שני שקלים, עבור מכתב גדול יש תוספת של שלושה שקלים

המחלקה `Package` מייצגת חבילה. בעבור כל מכתב יש לשמור את הנתונים הבאים:

- שם שולח החבילה, מטיפוס מחרוזת, `string`
- מען המכתב (כתובת), מטיפוס מחרוזת, `string`
- משקל החבילה בקילוגרמים מטיפוס מספר שלם, `int`
- האם שביר?: `true` – אם תוכן החבילה שביר, `false` – אם לא שביר.
- מחיר בסיסי למשלוח חבילה מטיפוס מספר ממשי, `double`

מחיר משלוח חבילה תלוי בגודל החבילה וסוגה: בעבור משקל פחות מחמישה קילוגרמים יש לשלם מחיר בסיסי. מעל חמישה קילוגרמים יש לשלם תוספת של שלושה שקלים <u>לכל קילוגרם נוסף</u>. אם התוכן שביר יש להוסיף עוד 15 ₪.

המחלקה `MailItem` היא מחלקת האב למחלקות `Letter` ו-`Package` (המחלקות `Letter` ו-`Package` יורשות מהמחלקה `MailItem`) בהתאם לעקרונות תכנות מונחה עצמים:



א. **(6 נק׳)** כתבו כותרות ותכונות של כל אחת מהמחלקות `MailItem`, `Letter` ו-`Package`

ב. **(4 נק׳)** כתבו פעולה `public double GetRealPrice()`.

הפעולה תחשב ותחזיר מחיר סופי של משלוח. יש לציין באיזה מחלקה/מחלקות צריך להוסיף את הפעולה בהתאם לעקרונות תכנות מונחה עצמים.

ג. **(6 נק׳)** כתבו פעולה

```
public static string BigSender(MailItem[] args)
```

הפעולה מקבלת מערך של משלוחים ומחזירה את שם השולח של החבילה היקרה ביותר (חבילה שמחיר המשלוח הוא הגבוה ביותר). אם אין אף חבילה במערך, הפעולה תחזיר ערך `null`.

אפשר להניח שיש רק משלוח אחד כזה. אפשר להניח שקיימות פעולות `Set/Get` בכל מחלקה.

<u>שאלה 5</u>

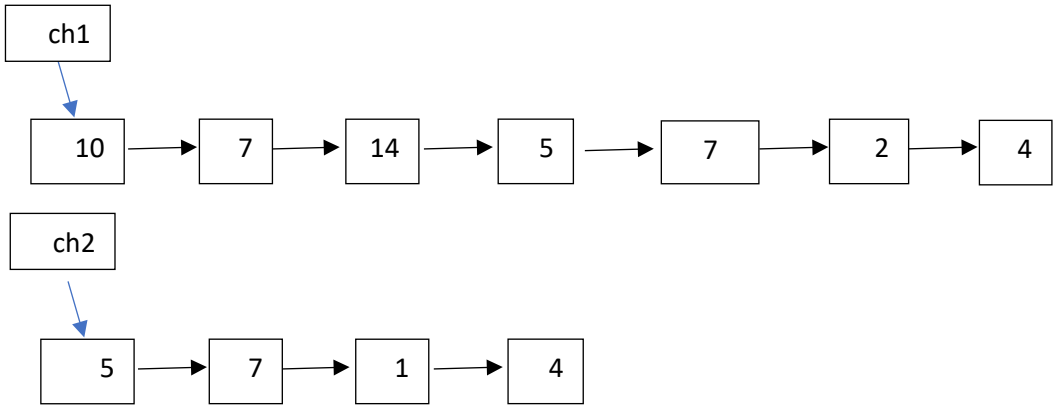
א. **(1 נק')** נתונה הפעולה `What` הבאה, מה מבצעת הפעולה?

```
public static int What(Node<int> ch){
        int c = 0;
        while(ch != null)
        {
            c++;
            ch = ch.GetNext();
        }
        return c;
    }
```

ב. **(4 נק')** נתונה הפעולה `Why` המשתמשת בפעולה `What`, מה תחזיר הפעולה `Why(ch1, ch2)`?

עבור שתי שרשרות החוליות הבאות: **יש להראות מעקב אחרי ביצוע הפעולה `Why`**.

אין צורך במעקב אחרי **`What`**.




```
public static bool Why(Node<int> ch1, Node<int> ch2) {
        int len1 = What(ch1);
        int len2 = What(ch2);
        for(int i = 0; i < len1-len2; i++)
        {
            ch1 = ch1.GetNext();
        }
        while(ch1!=null)
        {
            if(ch1.GetValue() != ch2.GetValue())
                return false;
            ch1 = ch1.GetNext();
            ch2 = ch2.GetNext();
        }
        return true;
 }
```

ג. **(4 נק')** תנו דוגמה של שרשרת `ch3` באורך של **ארבע** חוליות, באופן שבו זימון הפעולה `Why(ch1, ch3)` יחזיר ערך שונה מזה שהוחזר בסעיף ב'.

ד. **(4 נק')** תנו דוגמה של שרשרת `ch4` באורך של **ארבע** חוליות, באופן שבו זימון הפעולה `Why(ch4, ch1)` יחזיר ערך `true`. אם אי אפשר ליצור שרשרת כזאת, הסבירו למה.

ה. **(3 נק')** מה מבצעות הפעולות `What(ch)` ו- `Why(ch1,ch2)` באופן כללי?

חלק ב'

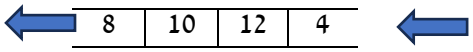
ענו על <u>שתיים</u> מבין השאלות 6-10 (ערך כל שאלה – 18 נקודות).

<u>שאלה 6</u>

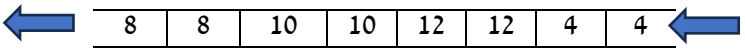
א. (7 נק')

כתבו פעולה `void One(Queue<int> q, int k)` המקבלת תור של מספרים שלמים q ומספר שלם וחיובי k. הפעולה "תכפיל" את התור באופן הבא:

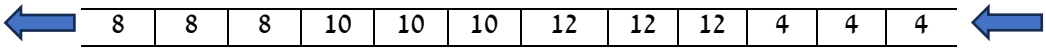
אם לפני זימון הפעולה התור q היה כך:



אחרי זימון הפעולה `One(q, 2)` התור q יהיה כך:



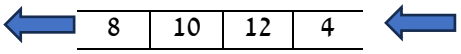
אחרי זימון הפעולה `One(q, 3)` התור q יהיה כך:



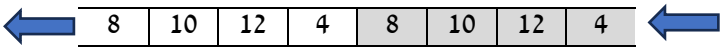
ב. (7 נק')

כתבו פעולה `void Two(Queue<int> q, int k)` המקבלת תור של מספרים שלמים q ומספר שלם וחיובי k. הפעולה "תכפיל" את תור בצורה הבאה:

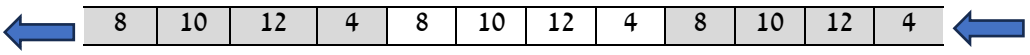
אם לפני זימון הפעולה התור q היה:



אז אחרי ביצוע זימון הפעולה `Two(q, 2)` התור q יהיה:



אז אחרי ביצוע זימון הפעולה `Two(q, 3)` התור q יהיה:



ג. **(4 נק')** מהן הסיבוכיות של הפעולות שכתבתם בסעיפים א'– ב'? **הסבירו את תשובתכם.**

שאלה 7

נתונה מחלקה המתארת אבן דומינו (Domino):

```
public class Domino
{
 private int left; // ערך מספרי צד שמאלי בין 0 ל-6 (כולל)
 private int right; // ערך מספרי צד ימני בין 0 ל-6 (כולל)

 public Domino(int left, int right)
 {
  this.left = left;
  this.right = right;
 }
}
```

במחלקה הוגדרו פעולות Set/Get.

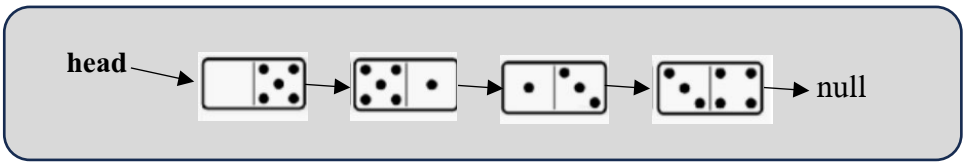
א. **(4 נק')** כתבו במחלקה Domino את הפעולה Rotate() . הפעולה מסובבת את האבן הנוכחית כלומר: מחליפה בין הערכים left ו- right.

נתונה המחלקה Deck המתארת רשימת אבני דומינו:

```
public class Deck
{
  private Node<Domino> head;
  public Deck()
  {
    this.head = null;
  }
}
```

רשימת אבני דומינו נקראת "מסודרת היטב" אם ערך של מספרי צד ימני של כל אבן (פרט לאבן האחרונה ברשימה) זהה לערך של מספרי צד שמאלי של האבן הבאה.

לדוגמה: הרשימה הבאה מוגדרת "מסודרת היטב"



רשימה ריקה או רשימה שיש בה רק אבן אחת, היא רשימה "מסודרת היטב".

ב. **(7 נק')** כתבו במחלקה Deck פעולה הבודקת שרשימה היא "מסודרת היטב". אם כן – הפעולה תחזיר ערך true, ולא – תחזיר ערך false.

ג. **(7 נק')** כתבו במחלקה Deck פעולה המקבלת אבן דומינו ובודקת אם אפשר להוסיף אותה לרשימה "מסודרת היטב" כך שהיא תישאר להיות "מסודרת היטב". אם כן – הפעולה תוסיף את האבן ותחזיר ערך true, ולא – תחזיר ערך false.

לדוגמה:

אפשר להוסיף לרשימה שבדוגמה בסעף א' את האבנים הבאות (0,4), (1,1), (4,6).

אי אפשר להוסיף את האבנים הבאות: (3,2), (5,6), (2,2).

<u>שאלה 8</u>

הגדרה: עץ **"מסודר"** הוא עץ בינארי של מספרים שלמים העונה על הכללים הבאים:

- לכל צומת פרט לעלים יש ערך זוגי
- הערכים של כל העלים הם מספרים אי-זוגיים
- הערך של צומת הבן גדול מהערך של צומת האב

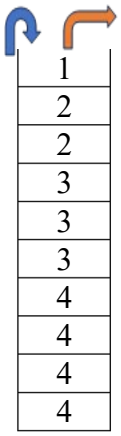
א. **(4 נק')** ציירו עץ "מסודר" הכולל לפחות שישה צמתים.

ב. **(14 נק')** כתבו פעולה המקבלת הפניה לשורש של עץ בינרי של מספרים שלמים. הפעולה תבדוק אם העץ הוא "עץ מסודר". אם כן – הפעולה תחזיר ערך `true`,
ולא – הפעולה תחזיר ערך `false`.

<u>שאלה 9</u>

מחסנית "פירמידה n" היא מחסנית של מספרים שלמים וחיוביים שהמספר הראשון (מספר שנימצא בראש המחסנית) הוא 1 וכל מספר נוסף מופיע בו ברצף, מספר פעמים השווה לערכו, עד N. כלומר, 1 יופיע פעם אחת, לאחריו 2 יופיע שתי פעמיים, מספר 3 מופיע שלוש פעמים וכו'.

לדוגמה: המחסנית "פירמידה 4":



א. **(7 נק')** כתבו פעולה המקבלת מספר שלם וחיובי **n** ומחזירה מחסנית "פירמידה n".

ב. **(7 נק')** כתבו פעולה המקבלת מחסנית של מספרים שלמים. הפעולה תבדוק אם המחסנית היא "פירמידה n". אם כן – הפעולה תחזיר ערך `true`, ולא – הפעולה תחזיר ערך `false`.

ג. (4 נק') מהן סיבוכיות הפעולות שכתבתם בסעיפים א' ו-ב'? **הסבירו את תשובתכם.**

שאלה 10

נתונות המחלקות A, B, C הבאות:

```
public class A
{
    private static int countA = 0;
    protected int x;
    public A()
     {
          countA++;
          this.x = 0;
    }
    public A(int x)
     {
          countA++;
          this.x = x;
    }
    public override string ToString()
     {
          return "A:"+this.x;
     }
    public virtual int DoIt(int z)
     {
          return this.x * z;
     }
    public static int GetCount(){ return countA;  }
}// end of class A

public class B : A
{
    protected int y;
    private static int countB = 0;
    public B(int y)
    {
          this.y = y;
          countB++;
    }
    public B(int x, int y):base(x)
    {
          countB++;
          this.y = y;
    }
    public override string ToString()
    {
        return "B:"+base.ToString ()+":"+this.y;
    }
    public static new int GetCount(){return countB;}
    public override int DoIt(int z)
    {
          return base.DoIt(z) + this.y;
     }
} //end of class B
```

```
public class C : B
{
    private A myA;
    private static int countC = 0;
    public C(int x, int y): base(x, y)
    {
        this.myA = new A(x+y);
        countC++;
    }
    public static new int GetCount(){return countC;}
    public override string ToString()
    {
        return "C:"+base. ToString()+" ["+this.myA+"]";
    }
}// end of class C
```

א. **(1 נק׳)** ציירו תרשים UML שמייצג את הקשרים בין המחלקות A, B, C.

יש לסמן ירושה באמצעות חץ ◄——— והכלה באמצעות הסימן ◆———

ב. **(5 נק׳)** נתון קטע הקוד הבא, בפעולה הראשית במחלקה אחרת:

```
A[] arr = new A[5];
arr[0] = new A(13);
arr[1] = new B(5, 10);
arr[2] = new C(2, 7);
arr[3] = new C(3, 3);
arr[4] = new B(5);
```

עקבו אחרי ביצוע קטע הקוד. יש לצייר את כל העצמים שנוצרו ולציין את ערכי התכונות שלהם.

ג. **(3 נק׳)** לקוד של סעיף ב׳ התווסף הקוד הבא, רשמו מה יהיה הפלט בהתאם:

```
Console.WriteLine("A objects: " + A.GetCount());
Console.WriteLine("B objects: " + B.GetCount());
Console.WriteLine("C objects: " + C.GetCount());
```

ד. **(5 נק׳)** לקוד של סעיף ב׳ התווסף הקוד הבא, רשמו מה יהיה הפלט:

```
for (int k=0; k <arr.Length; k++)
 Console.WriteLine(arr[k]);
```

ה. **(4 נק׳)** לקוד של סעיף ב׳ התווסף הקוד הבא, רשמו מה יהיה הפלט:

```
for (int k=0; k <arr.Length; k++)
 if(arr[k] is B)
     Console.WriteLine(arr[k].DoIt(k+1));
```

חלק ג׳

ענו על <u>אחת</u> מבין השאלות 11-12 (ערך שאלה – 16 נקודות).

<u>שאלה 11</u>

החברת **Helpl4U** נותנת שירותי תמיכה טלפונית. מוקד השירות כולל 15 עמדות תמיכה הממוספרות 0-14. שירותי התמיכה מתנהלים באופן הבא:

- כל מתקשר משאיר את שמו ואת מספר הטלפון שלו לרכז מערכת.
- הרכז מצרף את פרטי המתקשר לאוסף הממתינים בעמדת התמיכה הכי פחות עמוסה.
- כאשר עמדת התמיכה מסיימת טיפול במתקשר, פרטיו נמחקים מאוסף הממתינים של העמדה.

מערכת לניהול שירות כוללת כמה מחלקות.

המחלקה **Caller** (מתקשר). למחלקה הוגדרו פעולה בונה ופעולות `GetName()` ו- `GetPhOne()`

הפעולה בונה מתקשר חדש ששמו `name` ומספר הטלפון שלו `phOne`	`Caller (string name, string phOne)`

המחלקה **Service** מייצגת עמדת שירות. לפניכם ממשק חלקי של המחלקה `Service`:

מוסיפה מתקשר `c` לאוסף הממתינים בעמדה	**void** `AddCaller (Caller c)`
הפעולה מחזירה את המתקשר הראשון מבין הממתינים בעמדה, ומוחקת את נתוניו מאוסף הממתינים. **הנחה**: יש ממתינים בעמדה	**Caller** `Callback()`
הפעולה מחזירה את מספר ממתינים בעמדה	**int** `GetNumOfCallers()`

א. **(3 נק׳)** כתבו את כותרת המחלקה `Service` ואת התכונות שלה.

לייצוג אוסף, אפשר להשתמש במחלקות Stack, Queue, Node. אפשר להוסיף תכונות נוספות. חובה לתעד את התכונות.

ב. **(3 נק׳)** כתבו את הפעולה `Callback()`

ניהול מוקד השירות נעשה בעזרת המחלקה **Support**. לפניכם ממשק חלקי של המחלקה `Support`:

הפעולה מקבלת פרטים של מתקשר ומוסיפה אותו לסוף אוסף הממתינים של העמדה שמספר הממתינים בה הוא <u>הנמוך ביותר</u>. אם יש כמה עמדות כאלה, המתקשר יתווסף לראשונה מבניהן.	**void** `StartSupport (string name, string phOne)`

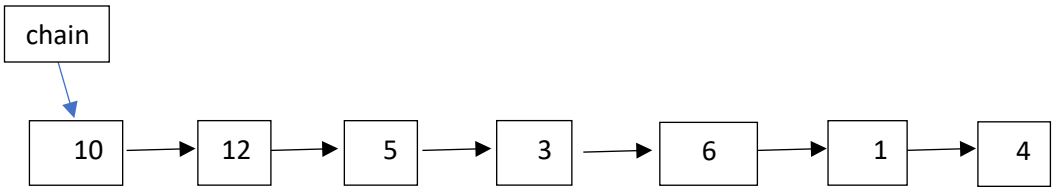
ג. **(3 נק׳)** כתבו את כותרת המחלקה `Support` ואת התכונות שלה

ד. **(3 נק׳)** כתבו את הפעולה `StartSupport()`

ה. **(4 נק׳)** הנהלת החברה החליטה לציין בסוף כל יום עבודה את פקיד/ת השרות שטיפל/ה בהכי הרבה לקוחות. ציינו שינוים הנדרשים במחלקה/מחלקות וכתבו במחלקה `Support` פעולה `TheBest()` המחזירה את מספר העמדה של הפקיד הטוב ביותר (אפשר להניח שיש רק פקיד אחד).

<u>שאלה 12</u>

נתונה השרשרת הבאה:



א. **(6 נק׳)** עקבו אחרי זימון הפעולה הרקורסיבית `One(chain)` הבאה וכתבו איך תיראה השרשרת אחרי ביצוע הפעולה? כמו כן, כתבו מה מבצעת הפעולה `One` באופן כללי?

יש להראות מעקב אחרי ביצוע הפעולה!

```
public static void One(Node<Integer> n){
    if(n.HasNext())
    {
        n.SetValue(n.GetNext().GetValue());
        One(n.GetNext());
    }
    else
    {
        n.SetValue(0);
    }
}
```

ב. **(6 נק׳)** נתונות הפעולה הרקורסיבית `Two` הבאה, עקבו אחרי זימון הפעולה `Two(chain,10)` ורשמו איך תיראה השרשרת אחרי ביצוע הפעולה? רשמו כמו כן מה מבצעת הפעולה באופן כללי?

יש להראות מעקב אחרי ביצוע הפעולה!

```
public static void Two(Node<Integer> n, int num){
    if(n.HasNext())
    {
        Two(n.GetNext(), num);
    }
    else
    {
        n.SetValue(num);
    }
}
```

ג. **(4 נק׳)** נתונה הפעולה `Three` הבאה, מה יהיה תוכן השרשרת `chain` אחרי זימון הפעולה `Three(chain)`? כתבו מה מבצעת הפעולה באופן כללי?

```
public static void Three(Node<Integer>n){
    int x = n.GetValue();
    One(n);
    Two(n,x);
}
```

בהצלחה!

נספח לשאלון 97105 – מבני נתונים ותכנות מונחה עצמים – JAVA

נספח ממשקים מבנה הנתונים בתוכנית הלימודים

ממשק המחלקה חוליה הגנרית- Node<T>

המחלקה מגדירה חוליה גנרית שבה ערך מטיפוס T והפניה לחוליה העוקבת.

הפעולה בונה חוליה. הערך של החוליה הוא x, ואין לה חוליה עוקבת.	`Node (T x)`
הפעולה בונה חוליה. הערך של החוליה הוא x, והחוליה העוקבת לה היא next. ערכו של next יכול להיות **null**.	`Node (T x, Node<T> next)`
הפעולה מחזירה את הערך של החוליה.	`T getValue()`
הפעולה מחזירה את החוליה העוקבת. אם אין חוליה עוקבת, הפעולה מחזירה **null**.	`Node<T> getNext()`
הפעולה משנה את הערך השמור בחוליה ל- x.	`void setValue (T x)`
הפעולה מחזירה **true** אם יש חוליה נוספת.	`boolean hasNext()`
הפעולה משנה את החוליה העוקבת ל- next. ערכו של next יכול להיות **null**.	`void setNext (Node<T> next)`
הפעולה מחזירה מחרוזת המתארת את החוליה.	`String toString()`

יעילות הפעולות: כל הפעולות מתבצעות בסדר גודל קבוע, O(1).

ממשק המחלקה הגנרית - מחסנית Stack<T>

המחלקה מגדירה טיפוס אוסף בעל פרוטוקול **LIFO** להכנסה והוצאה של ערכים.

Stack()	הפעולה בונה מחסנית ריקה.
boolean isEmpty()	הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אם היא אינה ריקה.
void push (T x)	הפעולה מכניסה את הערך x לראש המחסנית הנוכחית (דחיפה).
T pop()	הפעולה מוציאה את הערך שבראש המחסנית הנוכחית ומחזירה אותו (שליפה). **הנחה:** המחסנית הנוכחית אינה ריקה.
T top()	הפעולה מחזירה את הערך שבראש המחסנית הנוכחית מבלי להוציאו. **הנחה**: המחסנית הנוכחית אינה ריקה.
String toString()	הפעולה מחזירה תיאור של המחסנית, כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש המחסנית): $[x_1, x_2, \ldots, x_n]$

יעילות הפעולות- מחלקה מיוצגת בעזרת שרשרת חוליות.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה toString() המתבצעת בסדר גודל לינארי.

ממשק המחלקה הגנרית- תור Queue<T>

המחלקה מגדירה טיפוס אוסף עם פרוטוקול **FIFO** להכנסה והוצאה של ערכים.

Queue()	הפעולה בונה תור ריק.
boolean isEmpty()	הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ו"שקר" אם הוא אינו ריק.
void insert (Tx)	הפעולה מכניסה את הערך x לסוף התור הנוכחי.
T remove()	הפעולה מוציאה את הערך שבראש התור הנוכחי ומחזירה אותו. **הנחה**: התור הנוכחי אינו ריק.
T head()	הפעולה מחזירה את ערכו של האיבר שבראש התור מבלי להוציאו. **הנחה**: התור הנוכחי אינו ריק
String toString()	הפעולה מחזירה מחרוזת המתארת את התור כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש התור): $[x_1, x_2, \ldots, x_n]$

יעילות הפעולות- המחלקה מיוצגת בעזרת שרשרת חוליות והפניה לזנב התור.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה toString() המתבצעת בסדר גודל לינארי.

ממשק המחלקה הגנרית – חוליה בינרית<T>BinNode

המחלקה מגדירה חוליה בינרית שבה ערך מטיפוס T ושתי הפניות לחוליות בינריות.

הפעולה בונה חוליה בינרית. ערך החוליה הוא x וערך שתי ההפניות שלה הוא **null**	**BinNode** (T x)
הפעולה בונה חוליה בינרית שערכה יהיה x. left ו-right הן (הפניות אל) הילד השמאלי והימני שלה. ערכי ההפניות יכולים להיות **null**	BinNode (BinNode<T> left, T x, BinNode<T> right)
הפעולה מחזירה את הערך של החוליה	**T** getValue()
הפעולה משנה את הערך השמור בחוליה ל-x	**void** setValue (T x)
הפעולה מחזירה **true** אם יש ילד שמאלי	**boolean** hasLeft()
הפעולה מחזירה **true** אם יש ילד ימני	**boolean** hasRight()
הפעולה מחזירה את הילד השמאלי של החוליה. אם אין ילד שמאלי הפעולה מחזירה **null**	**BinNode<T>** getLeft()
הפעולה מחזירה את הילד הימני של החוליה. אם אין ילד ימני הפעולה מחזירה **null**	**BinNode<T>** getRight()
הפעולה מחליפה את הילד השמאלי בחוליה left	**void** setLeft (BinNode<T> left)
הפעולה מחליפה את הילד הימני בחוליה right	**void** setRight (BinNode<T> right)
הפעולה מחזירה מחרוזת המתארת את הערך השמור בחוליה	**String** toString()

יעילות הפעולות: כל הפעולות מתבצעות בסדר גודל קבוע, O(1).

נספח לשאלון 97105 – מבני נתונים ותכנות ונחה עצמים – C#

נספח ממשקים מבנה הנתונים בתוכנית הלימודים

ממשק המחלקה חוליה הגנרית- Node<T>

המחלקה מגדירה חוליה גנרית שבה ערך מטיפוס T והפניה לחוליה העוקבת.

הפעולה בונה חוליה. הערך של החוליה הוא x, ואין לה חוליה עוקבת.	**Node** (T x)
הפעולה בונה חוליה. הערך של החוליה הוא x, והחוליה העוקבת לה היא next. ערכו של next יכול להיות **null**.	Node (T x, Node<T> next)
הפעולה מחזירה את הערך של החוליה.	**T** GetValue()
הפעולה מחזירה את החוליה העוקבת. אם אין חוליה עוקבת, הפעולה מחזירה **null**.	**Node<T>** GetNext()
הפעולה משנה את הערך השמור בחוליה ל- x.	**void** SetValue (T x)
הפעולה מחזירה **true** אם יש חוליה נוספת.	**bool** HasNext()
הפעולה משנה את החוליה העוקבת ל-next. ערכו של next יכול להיות **null**.	**void** SetNext (Node<T> next)
הפעולה מחזירה מחרוזת המתארת את החוליה.	**override string** ToString()

יעילות הפעולות: כל הפעולות מתבצעות בסדר גודל קבוע, O(1).

ממשק המחלקה הגנרית - מחסנית Stack<T>

המחלקה מגדירה טיפוס אוסף בעל פרוטוקול **LIFO** להכנסה והוצאה של ערכים.

הפעולה בונה מחסנית ריקה.	**Stack**()
הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אם היא אינה ריקה.	**bool** IsEmpty()
הפעולה מכניסה את הערך x לראש המחסנית הנוכחית (דחיפה).	**void** Push (T x)
הפעולה מוציאה את הערך שבראש המחסנית הנוכחית ומחזירה אותו (שליפה). **הנחה**: המחסנית הנוכחית אינה ריקה.	**T** Pop()
הפעולה מחזירה את הערך שבראש המחסנית הנוכחית בלי להוציאו. **הנחה**: המחסנית הנוכחית אינה ריקה.	**T** Top()
הפעולה מחזירה תיאור של המחסנית, כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש המחסנית): $[x_1, x_2, \ldots, x_n]$	**override string** ToString()

יעילות הפעולות- מחלקה מיוצגת בעזרת שרשרת חוליות.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה () ToString המתבצעת בסדר גודל לינארי.

ממשק המחלקה הגנרית - תור Queue<T>

המחלקה מגדירה טיפוס אוסף עם פרוטוקול **FIFO** להכנסה והוצאה של ערכים.

הפעולה בונה תור ריק.	**Queue**()
הפעולה מחזירה "אמת" אם התור הנוכחי ריק, ו"שקר" אם הוא אינו ריק.	**bool** IsEmpty()
הפעולה מכניסה את הערך x לסוף התור הנוכחי.	**void** Insert (Tx)
הפעולה מוציאה את הערך שבראש התור הנוכחי ומחזירה אותו. **הנחה**: התור הנוכחי אינו ריק.	**T** Remove()
הפעולה מחזירה את ערכו של האיבר שבראש התור מבלי להוציאו. **הנחה**: התור הנוכחי אינו ריק.	**T** Head()
הפעולה מחזירה מחרוזת המתארת את התור כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש התור): $[x_1, x_2, \ldots, x_n]$	**override string** ToString()

יעילות הפעולות- המחלקה מיוצגת בעזרת שרשרת חוליות והפניה לזנב התור.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה () ToString המתבצעת בסדר גודל לינארי.

ממשק המחלקה חוליה בינרית BinNode <T>

המחלקה מגדירה חוליה בינרית שבה ערך מטיפוס T ושתי הפניות לחוליות בינריות.

הפעולה בונה חוליה בינרית. ערך החוליה הוא x וערך שתי ההפניות שלה הוא **null**	BinNode (**T** x)
הפעולה בונה חוליה בינרית שערכה יהיה x, left ו-right הן הפניות אל הילד השמאלי והימני שלה. ערכי ההפניות יכולים להיות **null**	BinNode (BinNode<**T**> left, T x, BinNode<T> right)
הפעולה מחזירה את הערך של החוליה	**T** GetValue()
הפעולה משנה את הערך השמור בחוליה ל-x	**void** SetValue (**T** x)
הפעולה מחזירה **true** אם יש ילד שמאלי	**bool** HasLeft()
הפעולה מחזירה **true** אם יש ימני	**bool** HasRight()
הפעולה מחזירה את הילד השמאלי של החוליה. אם אין ילד שמאלי הפעולה מחזירה **null**	**BinNode<T>** GetLeft()
הפעולה מחזירה את הילד הימני של החוליה. אם אין ילד ימני הפעולה מחזירה **null**	**BinNode<T>** GetRight()
הפעולה מחליפה את הילד השמאלי בחוליה left	**void** SetLeft (BinNode<**T**> left)
הפעולה מחליפה את הילד הימני בחוליה right	**void** SetRight (BinNode<**T**> right)
הפעולה מחזירה מחרוזת המתארת את הערך השמור בחוליה	**string** ToString()

יעילות הפעולות: כל הפעולות מתבצעות בסדר גודל קבוע, O(1)

מילון עזר – בחינת מה"ט 97105 – מועד ב' קיץ 24 – תשפ"ד

חלק א' - القسم أ

رقم السؤال	الكلمة \ التعبير بالعبرية	الكلمة \ الكلمة بالعربية
1	"שיוויון"	"مساواة"
2	אין מילים	لا توجد كلمات
3	שרשרת "חזרות"	سلسلة "مراجعات"
4	רשות הדואר	سلطة البريد
4	דואר רשום	بريد مُسجّل
4	מכתב בגודל סטנדרטי	مكتوب بحجم قياسي
4	משלוח\משלוחים	ارسالية/ارسليات
4	החבילה היקרה ביותר	الرزمة الاغلى
5	אין מילים	لا توجد كلمات

חלק ב' - القسم ب

رقم السؤال	الكلمة \ التعبير بالعبرية	الكلمة \ الكلمة بالعربية
6	אין מילים	لا توجد كلمات
7	אבן דומינו	حجر الدومينو
7	מסובבת	استدارة
7	"מסודרת היטב"	مُرتبة بشكل جيد
8	"מסודר"	"مُرتب"
9	פירמידה	هرم
10	אין מילים	لا توجد كلمات

חלק ג' - القسم ج

رقم السؤال	الكلمة \ التعبير بالعبرية	الكلمة \ الكلمة بالعربية
11	שירותי תמיכה טלפונית	خدمات الدعم الهاتفي
11	מוקד שירות	مركز الخدمة
11	עמדת תמיכה	موضع دعم
11	פקיד	موظف
12	אין מילים	لا توجد كلمات