משרד העבודה
MINISTRY OF LABOR

מה"ט
המכון הממשלתי להכשרה טכנולוגית

מספר ת"ז_____

מספר מחברת_____

מועד הבחינה: אביב תשפ"ה – 2025 – 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-C# (עמוד 2)

מבחן ב-JAVA (עמוד 14)

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 **אחד בלבד** עם **חומר ההרצאות**. 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 המבחן, ולוודא שהן מובנות.
2. את התשובות יש לכתוב בצורה מסודרת, בכתב יד ברור ונקי (גם בכך תלויה הערכת המבחן).
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 מחדר הבחינה!

בהצלחה!

מבחן ב-C#

חלק א׳

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

שאלה 1

(14 נק׳) א. מחסנית של מספרים שלמים נקראת ״מחסנית ייחודית״ אם היא מחסנית ריקה או מחסנית שכל אחד מהאיברים בה אינו מתחלק בשום איבר אחר.

כתבו פעולה המקבלת מחסנית S שאיבריה מספרים שלמים וחיוביים. אם המחסנית היא מחסנית ייחודית הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

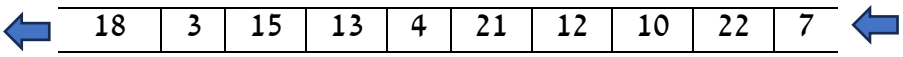
(2 נק׳) ב. מהי סיבוכיות הפעולה שכתבתם בסעיף א׳? **הסבירו את תשובתכם.**

שאלה 2

תור ״שווה סכומים״, הוא תור של מספרים שלמים המכיל מספר זוגי של איברים כך שסכומם של כל שני איברים קיצוניים (ראשון + אחרון, שני + לפני אחרון וכו׳) זהה.

לדוגמה:

התור הבא הוא ״**תור שווה סכומים**״



18+7 = 25 (האיבר הראשון והאיבר האחרון)

3+22 = 25 (האיבר השני והאיבר שלפני אחרון)

15+10 = 25

13+12 = 25

4+21 = 25

(14 נק׳) א. כתבו פעולה בשם `EqualSums` המקבלת תור של מספרים שלמים. אם התור הוא תור ״שווה סכומים״ הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר `false`.

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

שאלה 3

(14 נק׳) א. שרשרת חוליות של מספרים שלמים נקראת ״שרשרת מושלמת K״ אם כל ערך בה מופיע בדיוק K פעמים. כתבו פעולה המקבלת הפניה לחוליה הראשונה בשרשרת חוליות ומספר שלם וחיובי K. הפעולה תבדוק אם השרשרת היא ״שרשרת מושלמת K״. אם כן – הפעולה תחזיר ערך `true`, אם לא – הפעולה תחזיר ערך `false`.

כותרת הפעולה:

```
public static bool PerfectListK(Node<int> chain, int k)
```

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

<u>שאלה 4</u>

נתונות שתי המחלקות `Car, Motorcycle`. למחלקות יש תכונה אחת – מהירות מרבית.

```
public class Car {
    private int speed;
    public Car (int s) {
        speed = s;
    }

    public int GetSpeed () {
        return speed;
    }

    public bool Equals(Car other) {
        return (other!=null) && (speed == other.speed);
    }
}
public class Motorcycle {
    private int speed;

    public Motorcycle (int s) {
        speed = s;
    }

    public int GetSpeed () {
        return speed;
    }

    public bool Equals (Object other) {
        return ((other != null) &&
            (other is Motorcycle) &&
            (speed == ((Motorcycle)other).speed));
    }
}
```

(6 נק׳) א. כתבו פעולה המקבלת מערך הפניות לעצמים מסוג `Object`. הפעולה תחזיר מספר אופנועים (עצמים מסוג `Motorcycle`) שמהירותם המרבית היא בין 180 ל-250 קמ״ש.

כותרת הפעולה:

```
public static int CountCars(Object[] vehicles)
```

(10 נק׳) ב. נתונה המחלקה `Drive`:

```
public class Drive {
    public static void Main (string[] args) {
        Car c1 = new Car (180);
        Object c2 = new Car (180);
        Motorcycle m1 = new Motorcycle (180);
        Object m2 = new Motorcycle (180);
        //  *******
    }
}
```

בהתייחס לכל אחת מהשורות שלפניכם, כתבו מה יקרה בעקבות הוספתה לשיטה `Main` שלעיל, לאחר ההצהרות על האובייקטים (במקום שמסומן בכוכביות ***).
אם הוספת פקודת קוד גורמת לשגיאה יש להסביר מהי השגיאה (שגיאת קומפילציה או שגיאת זמן ריצה).

```
(1)     Console.WriteLine(c1.speed);
(2)     Console.WriteLine(((Motorcycle)c2).GetSpeed());
(3)     Console.WriteLine(c1.Equals(c2));
(4)     Console.WriteLine(c2.Equals(c1));
(5)     Console.WriteLine(m1.Equals(m2));
(6)     Console.WriteLine(m2.Equals(m1));
(7)     Console.WriteLine(c1.Equals((Motorcycle)m2));
(8)     Console.WriteLine(c1.Equals((Car)c2));
(9)     Console.WriteLine(m1.Equals((Car)c2));
(10)    Console.WriteLine(m1.Equals((Motorcycle)c2));
```

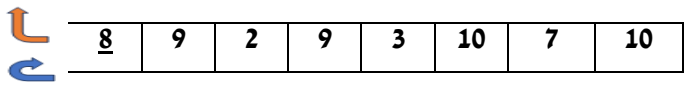
<u>שאלה 5</u>

נתונה הפעולה:

```
public static void What(Stack<int> st, bool flag)
{
    int x, y;
    bool found;
    Stack<int> temp1 = new Stack<int>();
    Stack<int> temp2 = new Stack<int>();
    while(!st.IsEmpty())
    {
        x = st.Pop();
        found = false;
        while(!st.IsEmpty())
        {
            y = st.Pop();
            if(x > y == flag)
                found = true;
            temp1.Push(y);
        }
        while(!temp1.IsEmpty())
            st.Push(temp1.Pop());

        if(!found)
            temp2.Push(x);
    }
    while(!temp2.IsEmpty())
        st.Push(temp2.Pop());
}
```

(7 נק׳) א. נתונה המחסנית st (8 נמצא בראש המחסנית):



<u>8</u>	**9**	**2**	**9**	**3**	**10**	**7**	**10**

מה יהיה התוכן של המחסנית st אחרי זימון הפעולה What(st, true) עבור המחסנית st?

יש להראות מעקב אחרי ביצוע הפעולה What.

(3 נק׳) ב. תנו דוגמה למחסנית st המכילה לפחות חמישה ערכים שונים כך שתוכן המחסנית לא ישתנה אחרי זימון הפעולה What(st, true).
אם לא ניתן ליצור מחסנית כזו, יש להסביר מדוע.

(3 נק׳) ג. תנו דוגמה למחסנית st המכילה לפחות חמישה ערכים שונים כך שאחרי זימון הפעולה What(st, false) במחסנית יישאר רק ערך אחד.
אם לא ניתן ליצור מחסנית כזו, יש להסביר מדוע.

(3 נק׳) ד. מה מבצעת הפעולה What(Stack<int> st, bool flag) באופן כללי?

חלק ב׳

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

<u>שאלה 6</u>

הערה כללית לשאלה:

- אין להשתמש במבני נתונים נוספים פרט לתור/תורים.
- יש לשמור את התור במצב מקורי בסיום הפעולה.
- אפשר להניח שקיימת הפעולה `Clone` המחזירה העתק של התור. כותרת הפעולה:

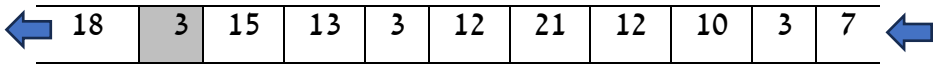
```
public static Queue<int> Clone(Queue<int> q)
```

סיבוכיות של הפעולה `Clone` היא `O(n)` כאשר `n` = מספר האיברים בתור.

(5 נק׳) א. כתבו פעולה המקבלת תור של מספרים שלמים `q` ומספר שלם `num`. הפעולה תחזיר את מיקומו **הראשון** של `num` בתוך התור. אם `num` לא מופיע בתור, הפעולה תחזיר ערך 1-.

לדוגמה:

עבור התור `q` הבא והמספר `num=3`



הפעולה תחזיר 2.

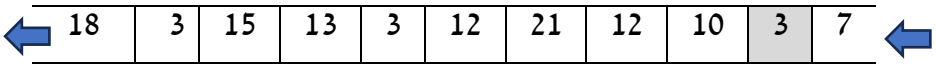
כותרת הפעולה:

```
public static int FirstPosition(Queue<int>q, int num)
```

(5 נק׳) ב. כתבו פעולה המקבלת תור של מספרים שלמים `q` ומספר שלם `num`. הפעולה תחזיר את מיקומו **האחרון** של `num` בתוך התור. אם `num` לא מופיע בתור, הפעולה תחזיר ערך 1-.

לדוגמה:

עבור התור `q` הבא והמספר `num=3`



הפעולה תחזיר 10.

כותרת הפעולה:

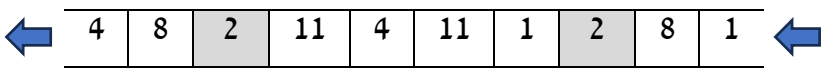
```
public static int LastPosition(Queue<int>q, int num)
```

(5 נק׳) ג. כתבו פעולה המקבלת מספר שלם וחיובי K ותור של מספרים שלמים, כל מספר בתור מופיע **פעמיים בדיוק**. הפעולה תבדוק אם קיים בתור זוג איברים זהים שמרחק ביניהם שווה ל-K.

ה**מרחק** בין שני איברים בתור הוא כמות האיברים הנמצאים ביניהם.

לדוגמה:

עבור התור הבא ו- K=4 הפעולה תחזיר `true` כי מרחק בין שני מספרים זהים (2 ו-2) הוא 4.



כותרת הפעולה:

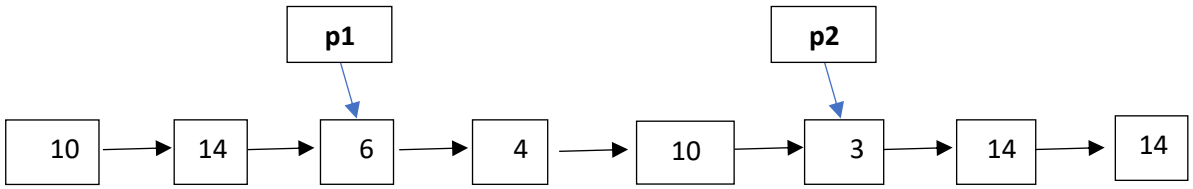
```
public static bool IsDistanceK(Queue<int> q, int k)
```

(3 נק׳) ד. מהי הסיבוכיות של הפעולות שכתבתם בסעיפים א׳–ג׳? **הסבירו את תשובתכם.**

שאלה 7

(5 נק׳) א. כתבו פעולה המקבלת שתי הפניות לחוליות שונות בשרשרת חוליות של מספרים שלמים. הפעולה תחזיר את סכום ערכי החוליות הנמצאות בין ההפניות שהתקבלו.

לדוגמה: עבור השרשרת שלפניכם הפעולה תחזיר 23.



כותרת הפעולה:

```
public static int GetSum(Node<int> p1, Node<int> p2)
```

אפשר להניח ש-`p1` מצביע על חוליה שנמצאת לפני החוליה ש-`p2` מצביע עליה.

(10 נק׳) ב. כתבו פעולה המקבלת הפניה לחוליה הראשונה של שרשרת חוליות של **מספרים שלמים חיוביים** ומספר שלם חיובי `num`. הפעולה תבדוק אם קיים רצף חוליות שסכום הערכים שלהן שווה ל-`num`. אם כן – הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

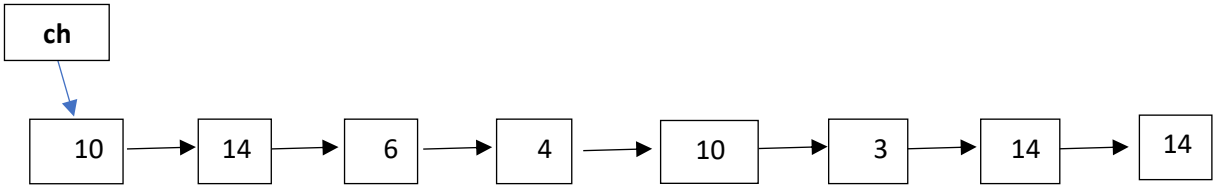
כותרת הפעולה:

```
public static bool IsAmount(Node<int> ch, int num)
```

לדוגמה:

עבור השרשרת הבאה ו-`num=23` או `num=37` הפעולה תחזיר `true`,

עבור `num=26` או `num=19` הפעולה תחזיר `false`.



(3 נק׳) ג. מהי הסיבוכיות של הפעולות שכתבתם בסעיפים א׳–ב׳? **הסבירו את תשובתכם.**

<u>שאלה 8</u>

(4 נק׳) א. כתבו פעולה המקבלת מחסנית של מספרים שלמים ומחזירה את סכום האיברים הנמצאים במחסנית.

כותרת הפעולה:

```
public static int GetSumOfStack(Stack<int> st)
```

(4 נק׳) ב. כתבו פעולה המקבלת מחסנית של מספרים שלמים ומחזירה את הערך של המספר האחרון שנמצא במחסנית (את הערך הנמצא בתחתית המחסנית).

כותרת הפעולה:

```
public static int GetStackLastValue(Stack<int> st)
```

שרשרת מחסניות היא שרשרת חוליות מסוג `Node<Stack<int>>`. כלומר, הערך (`value`) של כל חוליה הוא הפניה למחסנית של מספרים שלמים.

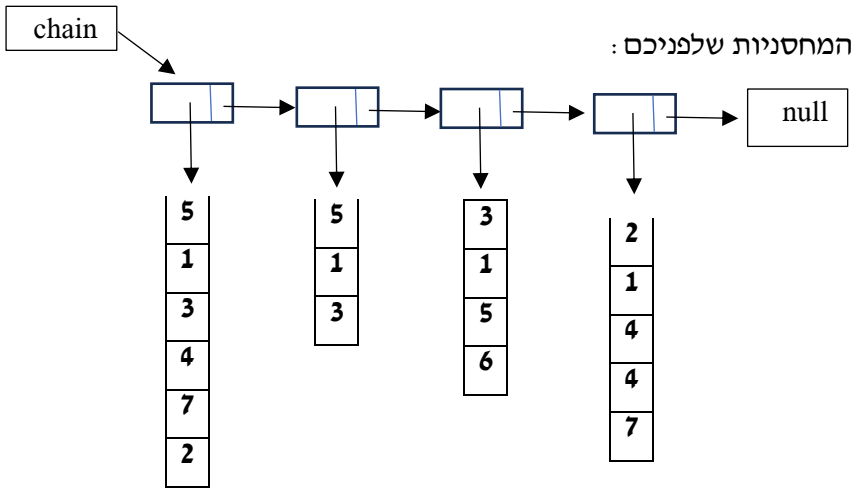
(7 נק׳) ג. כתבו פעולה המקבלת הפניה לחוליה הראשונה של שרשרת מחסניות.

הפעולה תחזיר תור של מספרים שלמים באופן הזה:

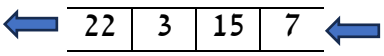
עבור כל מחסנית הנמצאת בשרשרת מחסניות שהאורך שלה **<u>זוגי</u>**, יוכנס לתור ערך השווה **לסכום** האיברים במחסנית, ואילו עבור כל מחסנית שהאורך שלה הוא **<u>אי-זוגי</u>** הפעולה תכניס לתור את הערך הנמצא בתחתית המחסנית.

לדוגמה,

עבור שרשרת המחסניות שלפניכם:



הפעולה תחזיר תור חדש:



- עבור החוליה הראשונה יתקבל **סכום** של איברים כי אורך המחסנית **זוגי**.
  הסכום הוא **22** = 5+1+3+4+7+2
- עבור החוליה השנייה יתקבל **מספר בתחתית המחסנית** (3) כי אורך המחסנית **אי-זוגי**.
- עבור החוליה השלישית יתקבל **סכום** של איברים כי אורך המחסנית **זוגי**.
  הסכום הוא **15** = 3+1+5+6.
- עבור החוליה השנייה יתקבל **מספר בתחתית המחסנית** (7) כי אורך המחסנית **אי-זוגי**.

כותרת הפעולה:

```
public static Queue<int> DoIt(Node<Stack<int>> chain)
```

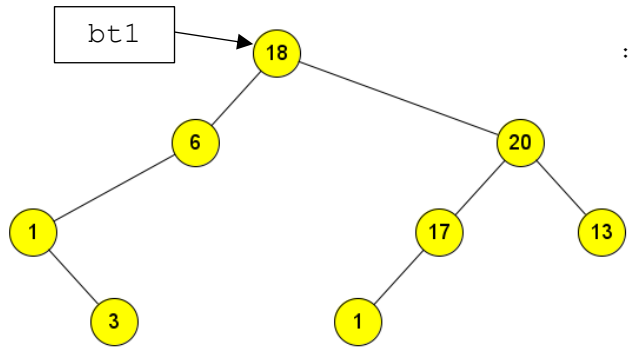
(3 נק׳) ד. מהי סיבוכיות זמן הריצה של הפעולה שכתבתם בסעיף ג׳? **הסבירו את תשובתכם**.

<u>שאלה 9</u>

נתונה הפעולה `Mystery`. הפעולה מקבלת הפניה לשורש עץ בינרי של מספרים שלמים חיוביים `bt`.

```
public static void Mystery(BinNode<int> bt)
{
    Mystery(bt, "");
}
private static void Mystery(BinNode<int> bt, string st)
{
    st = st + " " + bt.GetValue();
    if (bt.GetLeft()==null && bt.GetRight()==null)
     {
        Console.WriteLine(st);
     }
    else
     {
        if (bt.GetLeft() != null)
            Mystery (bt.GetLeft(), st);
        if (bt.GetRight() != null)
            Mystery (bt.GetRight(), st);
     }
}
```

נתון העץ הבינרי `bt1` הבא:



(6 נק׳) א. מה תהיה תוצאת זימון הפעולה `Mystery(bt1)`?

יש להראות מעקב אחרי ביצוע הפעולה `Mystery`.

(6 נק׳) ב. לפעולה `Mystery` הועבר כפרמטר הפניה לשורש של עץ בינרי `bt2`.
הפלט שהתקבל לאחר זימון הפעולה `Mystery(bt2)` היה בדיוק בסדר זה (משמאל לימין):

```
70  20  50  40  30
70  20  50  60
70  100 120
```

ציירו את העץ הבינרי `bt2` שהועבר כפרמטר לפעולה.

(3 נק׳) ג. לפעולה `Mystery` הועבר כפרמטר הפניה לשורש של **עץ חיפוש בינרי** `bt3`.
האם ייתכן שהפלט שהתקבל אחרי זימון הפעולה `Mystery(bt3)` יהיה זהה לפלט שהתקבל אחרי הזימון `Mystery(bt2)`? אם כן – ציירו **עץ חיפוש בינרי** `bt3`. אם הדבר בלתי אפשרי – הסבירו מדוע.

(3 נק׳) ד. מה מבצעת הפעולה `Mystery` באופן כללי?

שאלה 10

בבית החולים, לכל רופא מתמחה הוצמד רופא ותיק מנוסה. כדי לנהל את תהליך ההתמחות פותחו המחלקות האלה: `Doctor`, `Intern`, `Test`.

המחלקה **`Doctor`** מתארת רופא:

```
public class Doctor {
    protected string name;// שם
    protected string specialization;// התמחות
    protected int numOfPatients; // מספר חולים

    public Doctor(string name, string spec) {
        this.name = name;
        this.specialization = spec;
        this.numOfPatients = 0;
    }
    public Doctor(string name, string spec, int num) {
        this.name = name;
        this.specialization = spec;
        this.numOfPatients = num;
    }
    public Doctor(Doctor other) {
        this.name = other.name;
        this.specialization = other.specialization;
        this.numOfPatients = other.numOfPatients;
    }
    public void AddPatients(int num)
    {
       if(numOfPatients + num >=0)
          this.numOfPatients += num;
    }
    public string ToString(){
      return "Doctor:" + name + ", " + specialization+ ","
               +numOfPatients;
    }
}
```

במחלקה הוגדרו גם פעולות `set/Get` לכל התכונות.

המחלקה **`Intern`** מתארת מתמחה:

```
public class Intern : Doctor {
    private Doctor mentor;

    public Intern(string name, string spec, Doctor mentor):
                  base(name, spec){
        this.mentor = mentor;
        this.numOfPatients = mentor.numOfPatients/2;
    }
    public Doctor GetMentor(){return mentor;}
    public string ToString() {
       return "Intern: " + name + ", " + specialization +
          ", Mentor: " + mentor.name +", " +
          mentor.numOfPatients +"," + numOfPatients;
    }
}
```

(9 נק׳) א. נתונה הפעולה הראשית `main` אשר נמצאת במחלקה **`Test`**. עקבו אחרי הביצוע של הפעולה `main` ורשמו מה יהיו **ערכי התכונות של כל עצם שנוצר במהלך הביצוע**.

```
  public static void Main(string[] args) {
   Doctor[] d = new Doctor[9];
   d[0] = new Doctor("Dr. Cohen", "Cardiology",12);
   d[1] = new Doctor("Dr. Levy", "Neurology");
   d[2] = new Doctor("Dr. Sharon", "Pediatrics",8);

   d[3] = new Intern("Dani", "Cardiology", new Doctor(d[0]));
   d[4] = new Intern("Yael", "Surgery", d[0]);
   d[5] = new Intern("Avi", "Pediatrics", new Doctor(d[2]));
   d[6] = new Intern("Ruth", "Oncology", d[2]);
   d[7] = new Intern("Noam", "Cardiology", new Doctor(d[1]));
   d[8] = new Intern("Maya", "Neurology", new Doctor(d[0]));

   for (int i=0; I < d.Length; i++) {
      Console.WriteLine(d[i]);
   }

   d[0] = new Doctor("Dr. Goldman","Neurology",20);
   d[2].SetName("Dr. Galper");

   d[2].AddPatients(100);
   d[3].AddPatients(200);
   d[5].AddPatients(100);

   Console.WriteLine("After change:");

   for (int i=0; i < d.Length; i++) {
      Console.WriteLine(d[i]);
   }
}
```

(6 נק׳) ב. מהו הפלט של הפעולה `Main`?

(3 נק׳) ג. כתבו פעולה חיצונית המקבלת מערך רופאים ומדפיסה את שמותיהם של כל המתמחים אשר ההתמחות שלהם שונה מההתמחות של המנחים שלהם.

כותרת הפעולה:

```
public static void PrintNotSameSpec(Doctor[] d)
```

חלק ג׳

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

<u>שאלה 11</u>

מוקד החירום העירוני מקבל קריאות לעזרה בעקבות מצבי חירום שונים. בעת קבלת הקריאה נקבעת רמת הדחיפות של האירוע (1 – גבוהה, 2 – בינונית, 3 – נמוכה) וסוג האירוע (1 – שריפה, 2 – תאונה, 3 – חילוץ, 4 – פלילי, 5 – ביטחוני).

המחלקה **Call** מייצגת קריאה לעזרה.
תכונות המחלקה:

```
private int type; // סוג אירוע חירום
private string location; // אזור
private string description; // תיאור
private string callerName; // שם המתקשר
private int priority; // רמת הדחיפות
```

במחלקה הוגדרה פעולה בונה (constructor) המקבלת פרמטרים לכל תכונה וכן פעולות Get/Set.

המחלקה **EmergencyList** מייצגת רשימת קריאות לאירועי חירום.

(2 נק׳) א. כתבו את כותרת המחלקה **EmergencyList** ואת התכונות שלה.

לייצוג אוסף (רשימה) אפשר להשתמש במחלקות Stack, Queue, Node. אפשר להוסיף תכונות נוספות. יש לתעד את התכונות.

המחלקה **EmergencyCenter** מייצגת את מוקד החירום. למחלקה תכונה אחת – מערך של רשימות הקריאות לפי סוג אירוע החירום:

```
private EmergencyList[] arr = new EmergencyList[6];
```

התא 0 מיועד לרשימת קריאות שהטיפול בהן הסתיים.

(2 נק׳) ב. כתבו במחלקה **EmergencyCenter** פעולה המקבלת את הפרטים של הקריאה ומוסיפה אותה לרשימת קריאות <u>בהתאם לסוג האירוע</u> שלה.
כותרת הפעולה:

```
public void AddCall(int type, string location, string
                    description, string callerName, int priority)
```

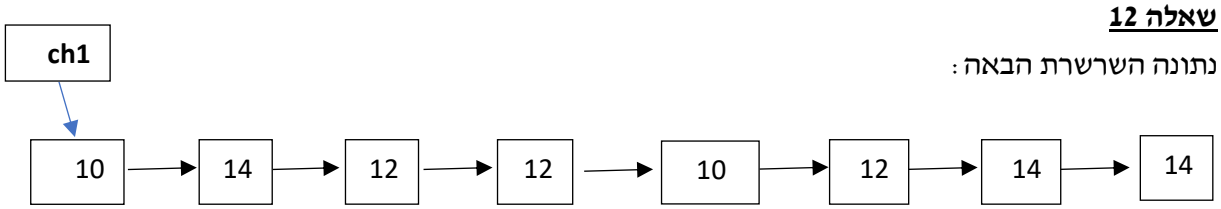
(6 נק׳) ג. כתבו במחלקה **EmergencyCenter** פעולה המקבלת סוג אירוע type ומחזירה קריאה לאירוע הדחוף ביותר מסוג זה. הפעולה גם תמחק את הקריאה מרשימת הקריאות ותעביר אותה לרשימת הקריאות שהטיפול בהן הסתיים.
אם יש מספר קריאות באותה רמת הדחיפות (הגבוהה ביותר), יש להחזיר את הקריאה שהתקבלה ראשונה. אם אין שום קריאה מסוג type, הפעולה תחזיר null.
כותרת הפעולה:

```
public Call MostUrgentCall(int type)
```

(6 נק׳) ד. כתבו במחלקה EmergencyCenter פעולה המקבלת אזור בעיר (location) ומדפיסה עבור כל סוג אירוע את מספר הקריאות באזור זה.

<u>שאלה 12</u>

נתונה השרשרת הבאה:



נתונות שתי הפעולות הרקורסיביות הבאות:

```
public static int One(Node<int> ch, int num)
{
    if(ch.GetNext()!= null)
    {
        if(ch.GetNext().GetValue() == num)
        {
            ch.SetNext(ch.GetNext().GetNext());
            return 1 + One(ch, num);
        }
        else
        {
            ch = ch.GetNext();
            return One(ch, num);
        }
    }
    else
        return 0;
}

public static void Two(Node<int> ch)
 {
    if(ch != null)
    {
        int x = One(ch, ch.GetValue());
        ch.SetNext(new Node<int> (x+1, ch.GetNext()));
        ch = ch.GetNext();
        Two(ch.GetNext());
    }
}
```

(6 נק׳) א. מה תהיה תוצאת זימון הפעולה `One(ch1, 14)`?

יש להראות מעקב ולציין שינוים בשרשרת החוליות בכל זימון הרקורסיבי

(2 נק׳) ב. מה מבצעת הפעולה `One` באופן כללי?

(6 נק׳) ג. עקבו אחרי זימון הפעולה `Two(ch1)` ורשמו איך תיראה השרשרת אחרי זימון.

יש להראות מעקב אחרי ביצוע הפעולה Two, אין צורך במעקב אחרי הזימונים של `One`.

(2 נק׳) ד. מה מבצעת הפעולה `Two` באופן כללי?

מבחן ב-JAVA

חלק א׳

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

שאלה 1

(14 נק׳) א. מחסנית של מספרים שלמים נקראת ״מחסנית ייחודית״ אם היא מחסנית ריקה או מחסנית שכל אחד מהאיברים בה אינו מתחלק בשום איבר אחר.

כתבו פעולה המקבלת מחסנית S שאיבריה מספרים שלמים וחיוביים. אם המחסנית היא מחסנית ייחודית הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

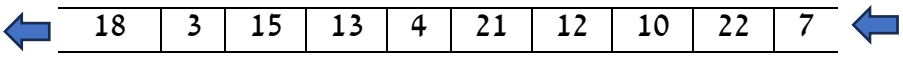
(2 נק׳) ב. מהי סיבוכיות הפעולה שכתבתם בסעיף א׳? **הסבירו את תשובתכם.**

שאלה 2

תור ״שווה סכומים״, הוא תור של מספרים שלמים המכיל מספר זוגי של איברים כך שסכומם של כל שני איברים קיצוניים (ראשון + אחרון, שני + לפני אחרון וכו׳) זהה.

לדוגמה:

התור הבא הוא **״תור שווה סכומים״**



18+7 = 25 (האיבר הראשון והאיבר האחרון)

3+22 = 25 (האיבר השני והאיבר שלפני אחרון)

15+10 = 25

13+12 = 25

4+21 = 25

(14 נק׳) א. כתבו פעולה בשם `equalSums` המקבלת תור של מספרים שלמים. אם התור הוא תור ״שווה סכומים״ הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר `false`.

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

שאלה 3

(14 נק׳) א. שרשרת חוליות של מספרים שלמים נקראת ״שרשרת מושלמת K״ אם כל ערך בה מופיע בדיוק K פעמים. כתבו פעולה המקבלת הפניה לחוליה הראשונה בשרשרת חוליות ומספר שלם וחיובי K. הפעולה תבדוק אם השרשרת היא ״שרשרת מושלמת K״. אם כן – הפעולה תחזיר ערך `true`, אם לא – הפעולה תחזיר ערך `false`.

כותרת הפעולה:

```
public static boolean perfectListK(Node<Integer> chain, int k)
```

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

שאלה 4

נתונות שתי המחלקות `Car, Motorcycle`. למחלקות יש תכונה אחת – מהירות מרבית.

```
public class Car {
    private int speed;
    public Car (int s) {
        speed = s;
    }

    public int getSpeed () {
        return speed;
    }

    public boolean equals(Car other) {
        return (other!=null) && (speed == other.speed);
    }
}
public class Motorcycle {
    private int speed;

    public Motorcycle (int s) {
        speed = s;
    }

    public int getSpeed () {
        return speed;
    }

    public boolean equals (Object other) {
        return ((other != null) &&
            (other instanceof Motorcycle) &&
            (speed == ((Motorcycle)other).speed));
    }
}
```

(6 נק׳) א. כתבו פעולה המקבלת מערך הפניות לעצמים מסוג `Object`. הפעולה תחזיר מספר אופנועים (עצמים מסוג Motorcycle) שמהירותם המרבית היא בין 180 ל-250 קמ״ש.

כותרת הפעולה:

```
public static int countCars(Object[] vehicles)
```

(10 נק׳) ב. נתונה המחלקה `Drive`:

```
public class Drive {
    public static void main (String[] args) {
        Car c1 = new Car (180);
        Object c2 = new Car (180);
        Motorcycle m1 = new Motorcycle (180);
        Object m2 = new Motorcycle (180);
        //  *******
    }
}
```

בהתייחס לכל אחת מהשורות שלפניכם, כתבו מה יקרה בעקבות הוספתה לשיטה `main` שלעיל, לאחר ההצהרות על האובייקטים (במקום שמסומן בכוכביות ***).
אם הוספת פקודת קוד גורמת לשגיאה יש להסביר מהי השגיאה (שגיאת קומפילציה או שגיאת זמן ריצה).

```
(11)    System.out.println (c1.speed);
(12)    System.out.println (((Motorcycle)c2).getSpeed());
(13)    System.out.println (c1.equals(c2));
(14)    System.out.println (c2.equals(c1));
(15)    System.out.println (m1.equals(m2));
(16)    System.out.println (m2.equals(m1));
(17)    System.out.println (c1.equals((Motorcycle)m2));
(18)    System.out.println (c1.equals((Car)c2));
(19)    System.out.println (m1.equals((Car)c2));
(20)    System.out.println (m1.equals((Motorcycle)c2));
```

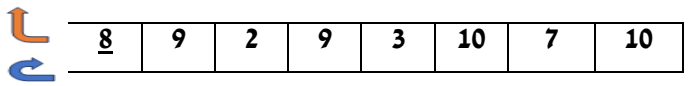
<u>שאלה 5</u>

נתונה הפעולה:

```
public static void what(Stack<Integer> st, boolean flag)
{
    int x, y;
    boolean found;
    Stack<Integer> temp1 = new Stack<Integer>();
    Stack<Integer> temp2 = new Stack<Integer>();
    while(!st.isEmpty())
    {
        x = st.pop();
        found = false;
        while(!st.isEmpty())
        {
            y = st.pop();
            if(x > y == flag)
                found = true;
            temp1.push(y);
        }
        while(!temp1.isEmpty())
            st.push(temp1.pop());

        if(!found)
            temp2.push(x);
    }
    while(!temp2.isEmpty())
        st.push(temp2.pop());
}
```

(7 נק׳) א. נתונה המחסנית st (8 נמצא בראש המחסנית):



<u>8</u>	**9**	**2**	**9**	**3**	**10**	**7**	**10**

מה יהיה התוכן של המחסנית st אחרי זימון הפעולה what(st, true) עבור המחסנית st?
יש להראות מעקב אחרי ביצוע הפעולה what.

(3 נק׳) ב. תנו דוגמה למחסנית st המכילה לפחות חמישה ערכים שונים כך שתוכן המחסנית לא ישתנה אחרי זימון הפעולה what(st, true).
אם לא ניתן ליצור מחסנית כזו, יש להסביר מדוע.

(3 נק׳) ג. תנו דוגמה למחסנית st המכילה לפחות חמישה ערכים שונים כך שאחרי זימון הפעולה what(st, false) במחסנית יישאר רק ערך אחד.
אם לא ניתן ליצור מחסנית כזו, יש להסביר מדוע.

(3 נק׳) ד. מה מבצעת הפעולה what(Stack<Integer> st, boolean flag) באופן כללי?

חלק ב׳

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

<u>שאלה 6</u>

הערה כללית לשאלה:

- אין להשתמש במבני נתונים נוספים פרט לתור/תורים.
- יש לשמור את התור במצב מקורי בסיום הפעולה.
- אפשר להניח שקיימת הפעולה `clone` המחזירה העתק של התור. כותרת הפעולה:

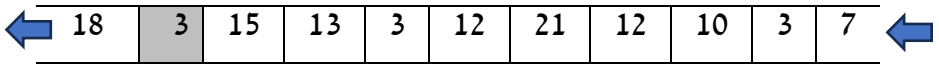
```
public static Queue<Integer> clone(Queue<Integer> q)
```

סיבוכיות של הפעולה `clone` היא `O(n)` כאשר `n` = מספר האיברים בתור.

(5 נק׳) א. כתבו פעולה המקבלת תור של מספרים שלמים `q` ומספר שלם `num`. הפעולה תחזיר את מיקומו **הראשון** של `num` בתוך התור. אם `num` לא מופיע בתור, הפעולה תחזיר ערך 1-.

לדוגמה:

עבור התור `q` הבא והמספר `num=3`



הפעולה תחזיר 2.

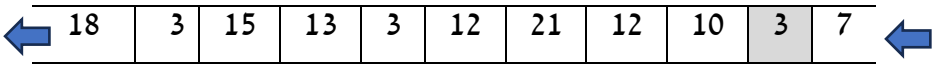
כותרת הפעולה:

```
public static int firstPosition(Queue<Integer>q, int num)
```

(5 נק׳) ב. כתבו פעולה המקבלת תור של מספרים שלמים `q` ומספר שלם `num`. הפעולה תחזיר את מיקומו **האחרון** של `num` בתוך התור. אם `num` לא מופיע בתור, הפעולה תחזיר ערך 1-.

לדוגמה:

עבור התור `q` הבא והמספר `num=3`



הפעולה תחזיר 10.

כותרת הפעולה:

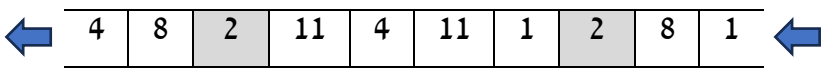
```
public static int lastPosition(Queue<Integer>q, int num)
```

(5 נק׳) ג. כתבו פעולה המקבלת מספר שלם וחיובי K ותור של מספרים שלמים, כל מספר בתור מופיע **פעמיים בדיוק**. הפעולה תבדוק אם קיים בתור זוג איברים זהים שמרחק ביניהם שווה ל-K.

ה**מרחק** בין שני איברים בתור הוא כמות האיברים הנמצאים ביניהם.

לדוגמה:

עבור התור הבא ו- K=4 הפעולה תחזיר `true` כי מרחק בין שני מספרים זהים (2 ו-2) הוא 4.



כותרת הפעולה:

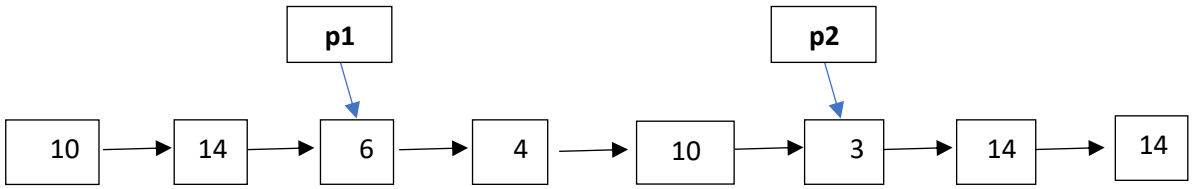
```
public static boolean isDistanceK(Queue<Integer> q, int k)
```

(3 נק׳) ד. מהי הסיבוכיות של הפעולות שכתבתם בסעיפים א׳–ג׳? **הסבירו את תשובתכם.**

שאלה 7

(5 נק׳) א. כתבו פעולה המקבלת שתי הפניות לחוליות שונות בשרשרת חוליות של מספרים שלמים. הפעולה תחזיר את סכום ערכי החוליות הנמצאות בין ההפניות שהתקבלו.

לדוגמה: עבור השרשרת שלפניכם הפעולה תחזיר 23.



כותרת הפעולה:

```
public static int getSum(Node<Integer> p1, Node<Integer> p2)
```

אפשר להניח ש-p1 מצביע על חוליה שנמצאת לפני החוליה ש-p2 מצביע עליה.

(10 נק׳) ב. כתבו פעולה המקבלת הפניה לחוליה הראשונה של שרשרת חוליות של **מספרים שלמים חיוביים** ומספר שלם חיובי num. הפעולה תבדוק אם קיים רצף חוליות שסכום הערכים שלהן שווה ל-num. אם כן – הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

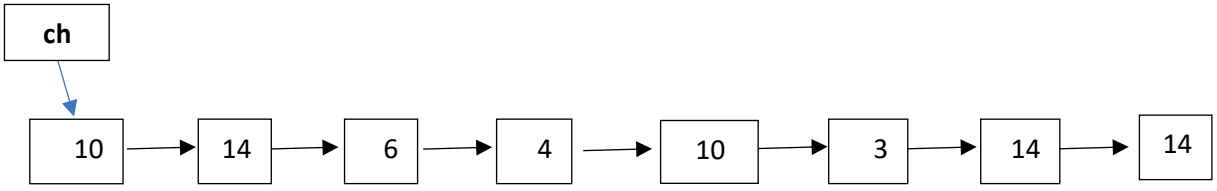
כותרת הפעולה:

```
public static boolean isAmount(Node<Integer> ch, int num)
```

לדוגמה:

עבור השרשרת הבאה ו-num=23 או num=37 הפעולה תחזיר true,

עבור num=26 או num=19 הפעולה תחזיר false.



(3 נק׳) ג. מהי הסיבוכיות של הפעולות שכתבתם בסעיפים א׳–ב׳? **הסבירו את תשובתכם.**

<u>שאלה 8</u>

(4 נק׳) א. כתבו פעולה המקבלת מחסנית של מספרים שלמים ומחזירה את סכום האיברים הנמצאים במחסנית.

כותרת הפעולה:

```
public static int getSumOfStack(Stack<Integer> st)
```

(4 נק׳) ב. כתבו פעולה המקבלת מחסנית של מספרים שלמים ומחזירה את הערך של המספר האחרון שנמצא במחסנית (את הערך הנמצא בתחתית המחסנית).

כותרת הפעולה:

```
public static int getStackLastValue(Stack<Integer> st)
```

שרשרת מחסניות היא שרשרת חוליות מסוג Node<Stack<Integer>>. כלומר, הערך (value) של כל חוליה הוא הפניה למחסנית של מספרים שלמים.

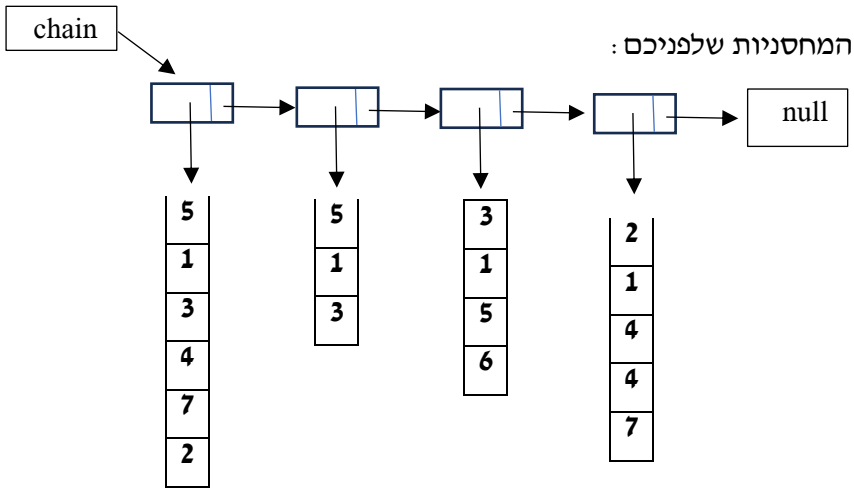
(7 נק׳) ג. כתבו פעולה המקבלת הפניה לחוליה הראשונה של שרשרת מחסניות.

הפעולה תחזיר תור של מספרים שלמים באופן הזה:

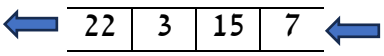
עבור כל מחסנית הנמצאת בשרשרת מחסניות שהאורך שלה **<u>זוגי</u>**, יוכנס לתור ערך השווה **לסכום** האיברים במחסנית, ואילו עבור כל מחסנית שהאורך שלה הוא **<u>אי-זוגי</u>** הפעולה תכניס לתור את הערך הנמצא בתחתית המחסנית.

לדוגמה,

עבור שרשרת המחסניות שלפניכם:



הפעולה תחזיר תור חדש:



- עבור החוליה הראשונה יתקבל **סכום** של איברים כי אורך המחסנית **זוגי**.
  הסכום הוא **22** = 5+1+3+4+7+2
- עבור החוליה השנייה יתקבל **מספר בתחתית המחסנית** (3) כי אורך המחסנית **אי-זוגי**.
- עבור החוליה השלישית יתקבל **סכום** של איברים כי אורך המחסנית **זוגי**.
  הסכום הוא **15** = 3+1+5+6.
- עבור החוליה השנייה יתקבל **מספר בתחתית המחסנית** (7) כי אורך המחסנית **אי-זוגי**.

כותרת הפעולה:

```
public static Queue<Integer> doIt(Node<Stack<Integer>> chain)
```

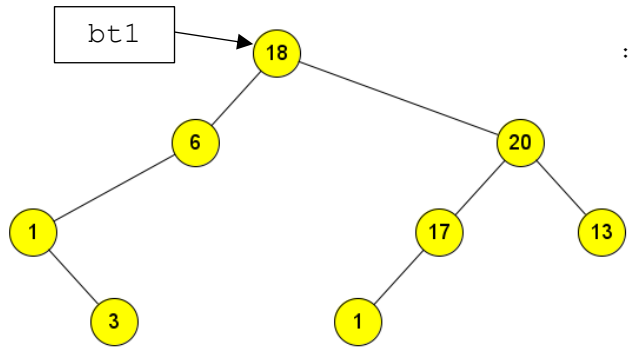
(3 נק׳) ד. מהי סיבוכיות זמן הריצה של הפעולה שכתבתם בסעיף ג׳? **הסבירו את תשובתכם**.

<u>שאלה 9</u>

נתונה הפעולה `mystery`. הפעולה מקבלת הפניה לשורש עץ בינרי של מספרים שלמים חיוביים `bt`.

```
public static void mystery(BinNode<Integer>bt)
{
    mystery(bt, "");
}
private static void mystery(BinNode<Integer> bt, String st)
{
    st = st + " " + bt.getValue();
    if (bt.getLeft()==null && bt.getRight()==null)
     {
        System.out.println(st);
     }
    else
     {
        if (bt.getLeft() != null)
            mystery (bt.getLeft(), st);
        if (bt.getRight() != null)
            mystery (bt.getRight(), st);
     }
}
```

נתון העץ הבינרי `bt1` הבא:



(6 נק׳) א. מה תהיה תוצאת זימון הפעולה `mystery(bt1)`?

יש להראות מעקב אחרי ביצוע הפעולה `mystery`.

(6 נק׳) ב. לפעולה `mystery` הועבר כפרמטר הפניה לשורש של עץ בינרי `bt2`.
הפלט שהתקבל לאחר זימון הפעולה `mystery(bt2)` היה בדיוק בסדר זה (משמאל לימין):

```
70  20  50  40  30
70  20  50  60
70  100 120
```

ציירו את העץ הבינרי `bt2` שהועבר כפרמטר לפעולה.

(3 נק׳) ג. לפעולה `mystery` הועבר כפרמטר הפניה לשורש של **עץ חיפוש בינרי** `bt3`.

האם ייתכן שהפלט שהתקבל אחרי זימון הפעולה `mystery(bt3)` יהיה זהה לפלט שהתקבל אחרי הזימון `mystery(bt2)`? אם כן – ציירו **עץ חיפוש בינרי** `bt3`. אם הדבר בלתי אפשרי – הסבירו מדוע.

(3 נק׳) ד. מה מבצעת הפעולה `mystery` באופן כללי?

<u>שאלה 10</u>

בבית החולים, לכל רופא מתמחה הוצמד רופא ותיק מנוסה. כדי לנהל את תהליך ההתמחות פותחו המחלקות האלה: `Doctor, Intern, Test`.

המחלקה **`Doctor`** מתארת רופא:

```
public class Doctor {
    protected String name;// שם
    protected String specialization;// התמחות
    protected int numOfPatients; // מספר חולים

    public Doctor(String name, String spec) {
        this.name = name;
        this.specialization = spec;
        this.numOfPatients = 0;
    }
    public Doctor(String name, String spec, int num) {
        this.name = name;
        this.specialization = spec;
        this.numOfPatients = num;
    }
    public Doctor(Doctor other) {
        this.name = other.name;
        this.specialization = other.specialization;
        this.numOfPatients = other.numOfPatients;
    }
    public void addPatients(int num)
    {
       if(numOfPatients + num >=0)
          this.numOfPatients += num;
    }
    public String toString(){
      return "Doctor:" + name + ", " + specialization+ ","
               +numOfPatients;
    }
}
```

במחלקה הוגדרו גם פעולות `set/get` לכל התכונות.

המחלקה **`Intern`** מתארת מתמחה:

```
public class Intern extends Doctor {
    private Doctor mentor;

    public Intern(String name, String spec, Doctor mentor) {
        super(name, spec);
        this.mentor = mentor;
        this.numOfPatients = mentor.numOfPatients/2;
    }
    public Doctor getMentor(){return mentor;}
    public String toString() {
       return "Intern: " + name + ", " + specialization +
          ", Mentor: " + mentor.name +", " +
          mentor.numOfPatients +"," + numOfPatients;
    }
}
```

(9 נק׳) א. נתונה הפעולה הראשית `main` אשר נמצאת במחלקה **`Test`**. עקבו אחרי הביצוע של הפעולה `main` ורשמו מה יהיו **ערכי התכונות של כל עצם שנוצר במהלך הביצוע**.

```
  public static void main(String[] args) {
   Doctor[] d = new Doctor[9];
   d[0] = new Doctor("Dr. Cohen", "Cardiology",12);
   d[1] = new Doctor("Dr. Levy", "Neurology");
   d[2] = new Doctor("Dr. Sharon", "Pediatrics",8);

   d[3] = new Intern("Dani", "Cardiology", new Doctor(d[0]));
   d[4] = new Intern("Yael", "Surgery", d[0]);
   d[5] = new Intern("Avi", "Pediatrics", new Doctor(d[2]));
   d[6] = new Intern("Ruth", "Oncology", d[2]);
   d[7] = new Intern("Noam", "Cardiology", new Doctor(d[1]));
   d[8] = new Intern("Maya", "Neurology", new Doctor(d[0]));

   for (int i=0; i<d.length; i++) {
      System.out.println(d [i]);
   }

   d[0] = new Doctor("Dr. Goldman","Neurology",20);
   d[2].setName("Dr. Galper");

   d[2].addPatients(100);
   d[3].addPatients(200);
   d[5].addPatients(100);

   System.out.println("After change:");

   for (int i=0; i<d.length; i++) {
      System.out.println(d[i]);
   }
}
```

(6 נק׳) ב. מהו הפלט של הפעולה `main`?

(3 נק׳) ג. כתבו פעולה חיצונית המקבלת מערך רופאים ומדפיסה את שמותיהם של כל המתמחים אשר ההתמחות שלהם שונה מההתמחות של המנחים שלהם.

כותרת הפעולה:

```
public static void printNotSameSpec(Doctor[] d)
```

חלק ג׳

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

<u>שאלה 11</u>

מוקד החירום העירוני מקבל קריאות לעזרה בעקבות מצבי חירום שונים. בעת קבלת הקריאה נקבעת רמת הדחיפות של האירוע (1 – גבוהה, 2 – בינונית, 3 – נמוכה) וסוג האירוע (1 – שריפה, 2 – תאונה, 3 – חילוץ, 4 – פלילי, 5 – ביטחוני).

המחלקה **Call** מייצגת קריאה לעזרה.
תכונות המחלקה:

```
private int type; // סוג אירוע חירום
private String location; // אזור
private String description; // תיאור
private String callerName; // שם המתקשר
private int priority; // רמת הדחיפות
```

במחלקה הוגדרה פעולה בונה (constructor) המקבלת פרמטרים לכל תכונה וכן פעולות get/set.

המחלקה **EmergencyList** מייצגת רשימת קריאות לאירועי חירום.

(2 נק׳) א. כתבו את כותרת המחלקה **EmergencyList** ואת התכונות שלה.

לייצוג אוסף (רשימה) אפשר להשתמש במחלקות Stack, Queue, Node. אפשר להוסיף תכונות נוספות. יש לתעד את התכונות.

המחלקה **EmergencyCenter** מייצגת את מוקד החירום. למחלקה תכונה אחת – מערך של רשימות הקריאות לפי סוג אירוע החירום:

```
private EmergencyList[] arr = new EmergencyList[6];
```

התא 0 מיועד לרשימת קריאות שהטיפול בהן הסתיים.

(2 נק׳) ב. כתבו במחלקה **EmergencyCenter** פעולה המקבלת את הפרטים של הקריאה ומוסיפה אותה לרשימת קריאות <u>בהתאם לסוג האירוע</u> שלה.
כותרת הפעולה:

```
public void addCall(int type, String location, String
                    description, String callerName, int priority)
```

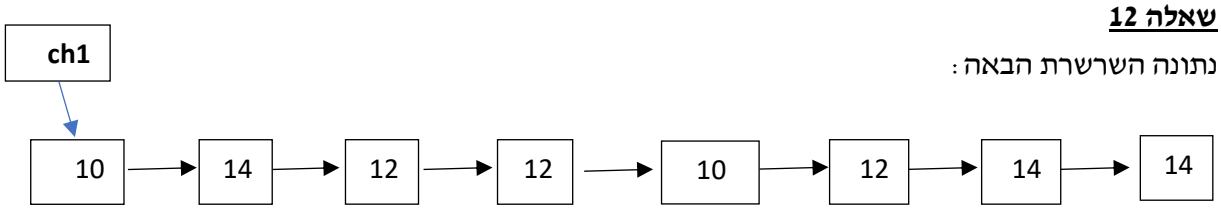
(6 נק׳) ג. כתבו במחלקה EmergencyCenter פעולה המקבלת סוג אירוע type ומחזירה קריאה לאירוע הדחוף ביותר מסוג זה. הפעולה גם תמחק את הקריאה מרשימת הקריאות ותעביר אותה לרשימת הקריאות שהטיפול בהן הסתיים.
אם יש מספר קריאות באותה רמת הדחיפות (הגבוהה ביותר), יש להחזיר את הקריאה שהתקבלה ראשונה. אם אין שום קריאה מסוג type, הפעולה תחזיר null.
כותרת הפעולה:

```
public Call mostUrgentCall(int type)
```

(6 נק׳) ד. כתבו במחלקה EmergencyCenter פעולה המקבלת אזור בעיר (location) ומדפיסה עבור כל סוג אירוע את מספר הקריאות באזור זה.

<u>שאלה 12</u>

נתונה השרשרת הבאה:



נתונות שתי הפעולות הרקורסיביות הבאות:

```
public static int one(Node<Integer> ch, int num)
{
    if(ch.getNext()!= null)
    {
        if(ch.getNext().getValue() == num)
        {
            ch.setNext(ch.getNext().getNext());
            return 1 + one(ch, num);
        }
        else
        {
            ch = ch.getNext();
            return one(ch, num);
        }
    }
    else
        return 0;
}

public static void two(Node<Integer> ch)
 {
    if(ch != null)
    {
        int x = one(ch, ch.getValue());
        ch.setNext(new Node<Integer> (x+1, ch.getNext()));
        ch = ch.getNext();
        two(ch.getNext());
    }
}
```

(6 נק׳) א. מה תהיה תוצאת זימון הפעולה `one(ch1, 14)`?

יש להראות מעקב ולציין שינוים בשרשרת החוליות בכל זימון הרקורסיבי

(2 נק׳) ב. מה מבצעת הפעולה `one` באופן כללי?

(6 נק׳) ג. עקבו אחרי זימון הפעולה `two(ch1)` ורשמו איך תיראה השרשרת אחרי זימון.

יש להראות מעקב אחרי ביצוע הפעולה `two`, אין צורך במעקב אחרי הזימונים של `one`.

(2 נק׳) ד. מה מבצעת הפעולה `two` באופן כללי?

נספח ממשקים מבנה הנתונים בתוכנית הלימודים-JAVA

ממשק החוליה 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.

הפעולה בונה מחסנית ריקה	**Stack<T>**()
הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אחרת	**boolean** 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 מבלי להוציאו. **הנחה**: המחסנית הנוכחית אינה ריקה	**T** top()
הפעולה מחזירה תיאור של המחסנית, כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש המחסנית): $[x_1, x_2, ..., x_n]$	**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<T>**()
הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ו"שקר" אחרת	**boolean** 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): $[x_1, x_2, ..., x_n]$	**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 הגנרית - 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 חוליה משמאל	**boolean** hasLeft()
הפעולה מחזירה **true** אם יש חוליה מימין	**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 ממשקים מבנה הנתונים בתוכנית הלימודים - C#

ממשק החוליה 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	**string** ToString()

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

ממשק המחלקה מחסנית STACK<T>

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

הפעולה בונה מחסנית ריקה	Stack<T>()
הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אחרת	**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 מבלי להוציאו. **הנחה**: המחסנית הנוכחית אינה ריקה	**T** Top()
הפעולה מחזירה תיאור של המחסנית, כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש המחסנית): $[x_1, x_2, ..., x_n]$	**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 תור QUEUE<T>

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

הפעולה בונה תור ריק	Queue<T>()
הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ו"שקר" אחרת	**bool** IsEmpty()
הפעולה מכניסה את הערך x לסוף התור הנוכחי	**void** Insert (**T** x)
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): $[x_1, x_2, ..., x_n]$	**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 חוליה משמאל	**bool** HasLeft()
הפעולה מחזירה **true** אם יש חוליה מימין	**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)