משרד העבודה
MINISTRY OF LABOR

מה"ט
המכון הממשלתי להכשרה טכנולוגית

מספר ת"ז_____
מספר מחברת_____

מבנה נתונים ותכנות מונחה עצמים

הנדסאים וטכנאים – הנדסת תוכנה

הנחיות לבחינה

מועד הבחינה: אביב תשפ"ו – 2026 – מועד א'
מספר השאלון: 97105
נספח ממשקים לבחינה JAVA
נספח ממשקים לבחינה C#
מילון עזר

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 12)

בכל מבחן תשע שאלות.

חלק א' – 32 נקודות

שאלות 1–3: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 16 נקודות.

חלק ב' – 32 נקודות

שאלות 4–6: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 16 נקודות.

חלק ג' – 36 נקודות

שאלות 7–9: יש לענות על **שתי** שאלות בלבד. ערך כל השאלה 18 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#

הנחיות כלליות לנבחנים:

בנספחי הבחינה מובאים ממשקים (interface) הכוללים פונקציות ומחלקות עזר, יש להקפיד ולהשתמש בהן בשאלות הרלוונטיות בבחינה על מנת להימנע מאיבוד ניקוד.

מבחן ב-C#

חלק א׳

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

<u>שאלה 1</u>

הגדרה:

סדרת מספרים נקראת ״עולה – יורדת״ אם עד מספר מסוים המספרים הולכים וגדלים וממספר זה ואילך, הם הולכים וקטנים.

לדוגמה, הסדרה שלפניכם היא סדרה ״עולה – יורדת״:

```
4, 9, 13, 20, 45, 33, 18, 1
```

(14 נק׳) א. כתבו פעולה המקבלת מחסנית של מספרים שלמים. הפעולה תבדוק אם הערכים שבמחסנית מהווים סדרה ״עולה – יורדת״. אם כן – הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

4
9
13
20
45
33
18
1

יש לשמור על שלמות המחסנית בסוף הפעולה.

כותרת הפעולה:

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

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

<u>שאלה 2</u>

(6 נק׳) א. כתבו פעולה המקבלת הפניה לחוליה הראשונה בשרשרת חוליות של מספרים שלמים ומספר שלם חיובי `pos`. הפעולה תחזיר איבר שנמצא במקום `pos` בשרשרת. אם אין מקום `pos` בשרשרת, הפעולה תחזיר ערך (-1).

יש להניח כי המיקום נספר החל מ-1, כאשר `pos = 1` מתייחס לחוליה הראשונה.

```
public static int ValueAt(Node<int> ch, int pos)
```

(8 נק׳) ב. כתבו פעולה המקבלת שתי הפניות לחוליות הראשונות של שתי שרשראות של מספרים שלמים `ch1` ו – `ch2`. הפעולה תיצור שרשרת חדשה לפי הכלל הבא:

- ראשית מכניסים את האיבר הראשון מהשרשרת `ch1`, ואחריו – את האיבר האחרון מהשרשרת `ch2`;
- מכניסים את האיבר השני מהשרשרת `ch1`, ואחריו את האיבר שלפני האחרון מהשרשרת `ch2`;
- וכך הלאה.

הפעולה תחזיר הפניה לחוליה הראשונה של השרשרת החדשה.

<u>אפשר להניח שאורכן של שתי השרשראות שווה.</u>

כותרת הפעולה:

```
public static Node<int> Merge (Node<int>ch1, Node<int>ch2)
```

(2 נק׳) ג. מהן הסיבוכיות של הפעולות שכתבתם בסעיפים א׳ ו-ב׳? **הסבירו את תשובתכם.**

שאלה 3

בפרויקט פיתוח רכב אוטונומי הוגדרו שלושה ממשקים ושתי מחלקות:

```
public interface Steering {
    void TurnLeft();
    void TurnRight(int degrees);
}
public interface Navigation{ void SetDestination(string dest); }
public interface Sensing { void ActivateLidar(); }
public class AutonomousCar : Steering, Navigation, Sensing {
    private string modelName;
    private double batteryLevel;
    // ...
}
public class TestDrive {
    public static void Main(string[] args) {
        AutonomousCar car1 = new AutonomousCar("Tesla Y", 98.5);
        AutonomousCar car2 = new AutonomousCar("Waymo One");
        // ( *****)
    }
}
```

(4 נק׳) א. השלימו את המחלקה `AutonomousCar`: כתבו כותרות לפעולות החסרות בה, כולל כל הפעולות הבונות (`constructors`) הנדרשות כדי שהקוד בפעולה `Main` יהיה תקין, וכן את כל הפעולות הנדרשות למימוש הממשקים. **אין צורך לממש את גוף הפעולות**.

(12 נק׳) ב. בכל אחד מסעיפים הבאים השורה (*****) תוחלף בשורת קוד משורות הקוד 1–6 מהטבלה הבאה. עבור כל אחד מהסעיפים, כתבו אם הקוד תקין או לא. אם הקוד אינו תקין – יש להסביר למה הוא אינו תקין , יש לציין את סוג השגיאה (קומפילציה או זמן ריצה).

שימו לב: אין קשר בין סעיפים.

1	`Steering s1 = car1;` `s1.TurnRight(45);`
2	`Navigation n2 = new Navigation();` `n2.SetDestination("Tel Aviv");`
3	`Navigation n3 = new AutonomousCar("Audi A8", 60.0);` `Steering s3 = (AutonomousCar)n3;`
4	`Navigation n4 = new AutonomousCar("Mercedes");` `n4.ActivateLidar();`

5	`Steering s5 = new AutonomousCar("Ford");`
	`Sensing sn5 = (Sensing) s5 ;`
	`sn5.ActivateLidar();`

6	`Sensing sn6 = new AutonomousCar("Nissan", 70.0);`
	`((AutonomousCar) sn6).TurnLeft();`
	`((Navigation) sn6).SetDestination("Haifa");`

חלק ב'

ענו על <u>שתיים</u> מבין השאלות 4–6 (ערך כל שאלה – 16 נקודות).

<u>שאלה 4</u>

(6 נק') א.

הגדרה:

שני מספרים שלמים וחיוביים num1 ו-num2 נקראים "זרים זה לזה" אם אין להם שום ספרה משותפת.

לדוגמה:

המספרים 81224 ו-973656 הם זרים, וגם המספרים 1000 ו-289 הם זרים.

המספרים 14 ו-21 הם לא זרים כי יש להם ספרה משותפת, 1.

כתבו פעולה המקבלת שני מספרים שלמים חיוביים. הפעולה תבדוק אם הם "זרים". אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

כותרת הפעולה:

```
public static bool AreStrangers(int num1, int num2)
```

(7 נק') ב.

כתבו פעולה המקבלת תור של מספרים שלמים חיוביים. הפעולה תבדוק אם כל הערכים בתור זרים זה לזה. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

כותרת הפעולה:

```
public static bool IsStrangersQueue(Queue<int> q)
```

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

שאלה 5

חברת סטארט-אפ פיתחה פלטפורמה לשירותי רשת. הטבלה שלפניכם מייצגת את היררכיית היישומים (Applications) במערכת.

מחלקה	יורשת מ-	תיאור
Application	Object	מחלקת אב – יישום כללי במערכת
WebService	Application	שירות רשת (Web)
MobileApp	Application	אפליקציה לנייד
BackendService	WebService	שירות צד שרת
FrontendApp	WebService	שירות צד לקוח
DataQuery	MobileApp	כלי לשליפת נתונים באפליקציה ניידת

(2 נק׳) א. שרטטו את עץ הירושה הכולל של המחלקות (Class Diagrams).

(8 נק׳) ב. נתון קטע קוד מתוך הפעולה הראשית (main):

```
Application a1 = new Application();
Application a2 = new FrontendApp();
Application a3 = new MobileApp();
WebService ws1 = new BackendService();
MobileApp ma1 = new DataQuery();
```

לפניכם שמונה קטעי קוד. עבור כל אחד מהסעיפים יש לציין אם הקוד תקין או לא. אם הקוד אינו תקין – יש להסביר למה הוא אינו תקין ולציין את סוג השגיאה (שגיאת קומפילציה/הידור או שגיאת זמן ריצה).

שימו לב: אין קשר בין הקטעים.

1	`WebService ws2 = new Application();`
2	`WebService ws3 = a2;`
3	`WebService ws4 = (WebService)a2;`
4	`WebService ws5 = (WebService)a3;`
5	`BackendService bs1 = (BackendService)ws1;`
6	`MobileApp ma2 = (MobileApp)a1;`
7	`Application a4 = (MobileApp)(new DataQuery());`
8	`FrontendApp fa1 = (WebService)(new BackendService());`

(6 נק׳) ג. לכל שירות במערכת יש עלות תפעול משלו (Cost). המטרה היא לחשב את סכום העלויות של כל היישומים שהם שירותי הרשת (WebService).

1. באילו מהמחלקות יש להגדיר את הפעולה `public double GetCost()` המחזירה את עלות התפעול? **הסבירו את תשובתכם**.
2. כתבו פעולה סטטית בשם SumWebServiceCost המקבלת מערך של יישומים ומחזירה את סכום העלויות של כל **שירותי הרשת**. כותרת הפעולה:

```
public static double SumWebServiceCost(Application[] arr)
```

הניחו שהפעולה GetCost() מומשה כראוי .

שאלה 6

נתונה המחלקה SpecNode:

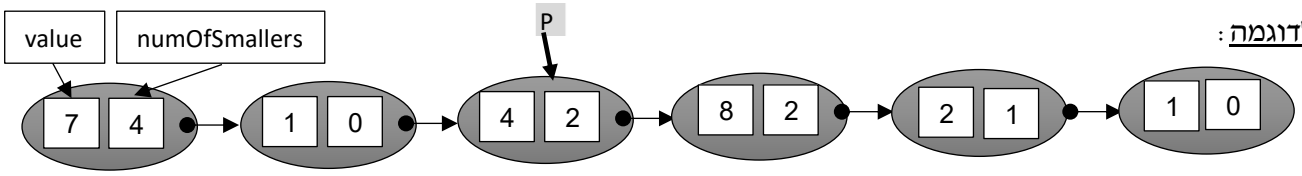
```
public class SpecNode {
    private int value;
    private int numOfSmallers;
    private SpecNode next;

   public SpecNode (int val) {
      this.value = val;
      this. numOfSmallers = 0;
      this.next = null;
    }
}
```

לכל תכונה במחלקה הוגדרו פעולות Get/Set.

בשרשרת חוליות מסוג SpecNode:

- value – מספר שלם.
- numOfSmallers – מספר החוליות המופיעות **אחרי** החוליה הנוכחית בשרשרת שערך ה-value שלהן **קטן** מערך ה-value של החוליה הנוכחית.
- next – מצביע לחוליה הבאה.

לדוגמה:



- ערך ה-value של החוליה הראשונה הוא 7, ואחריה יש ארבע חוליות שערכי ה-value שלהן קטן מ-7 ובגלל זה ערך התכונה numOfSmallers של החוליה הראשונה הוא 4.
- P מצביע על חוליה עם value=4, והתכונה numOfSmallers=2 מציינת כי יש שתי חוליות אחריה שערכי ה-value שלהן קטנים מ-4 (כלומר 1 ו-2).

(14 נק׳) א.

כתבו פעולה חיצונית המכניסה חוליה חדשה לשרשרת במיקום pos ועם ערך val. הפעולה מקבלת הפניה לחוליה הראשונה של שרשרת חוליות מסוג SpecNode, מספר שלם val ומספר שלם חיובי pos. (המיקום הראשון בשרשרת הוא 1). **אפשר להניח כי מיקום pos חוקי (אינו עולה על אורך רשימה+1)**.

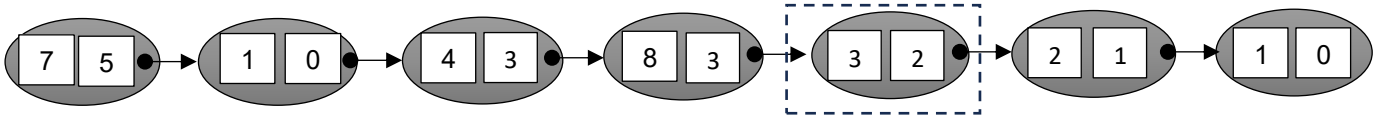
לאחר ההוספה, יש לעדכן את כל ערכי numOfSmallers בהתאם.

כותרת הפעולה:

```
public static SpecNode AddNum (SpecNode list, int val, int pos)
```

לדוגמה:

אחרי זימון הפעולה AddNum (list, 3, 5) השרשרת שתתקבל תהיה:



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

חלק ג'

ענו על <u>שתיים</u> מבין השאלות 7–9 (ערך שאלה – 18 נקודות).

<u>שאלה 7</u>

נתון פרויקט שהוגדרו בו שלוש מחלקות `Employee, Manager, Intern` ומחלקה `Driver`, כך:

```
public class Employee {
    private int seniority; // ותק

    public Employee(int seniority) {
        this.seniority = seniority;
    }
    public int GetSeniority() {
        return this.seniority;
    }
    public virtual bool Check(Employee e) {
        Console.WriteLine("EmpCheck");
        return this.seniority > e.seniority;
    }
}
```

```
public class Manager : Employee {
    public Manager(int seniority): base(seniority) {
    }
    public bool Check(Manager m) {
        Console.WriteLine("MgrCheck");
        return this.GetSeniority() == m.GetSeniority();
    }
}
```

```
public class Intern : Employee {
  private int mentorId;

  public Intern(int seniority, int mentorId): base(seniority){
        this.mentorId = mentorId;
  }
  public override bool Check(Employee e) {
      Console.WriteLine("IntCheck");
      if (!(e is Intern)) {
          return true;
      }
      Intern other = (Intern) e;
      return this.mentorId == other.mentorId;
  }
}
```

```
public class Driver
{
    public static void Main(string[] args)
    {
        Employee e1 = new Manager(15);
        Manager  m1 = new Manager(15);
        Employee e2 = new Employee(10);
        Employee i1 = new Intern(2, 800);
        Intern   i2 = new Intern(2, 900);
        Intern   i3 = new Intern(5, 800);


  (1)     Console.WriteLine(e1.Check(m1));
  (2)     Console.WriteLine(m1.Check(e1));
  (3)     Console.WriteLine(m1.Check(m1));
  (4)     Console.WriteLine(i1.Check(e2));
  (5)     Console.WriteLine(i1.Check(i2));
  (6)     Console.WriteLine(i1.Check(i3));
  (7)     Console.WriteLine(e2.Check(i1));

    }
  }
```

(14 נק׳) א. עבור כל אחת מהפקודות 1–7 כתבו מה יהיה הפלט של הפקודה.

(4 נק׳) ב. נתונה הפקודה: `Console.WriteLine(i1.Check(`**XXX**`)` כאשר **XXX** הוא משתנה שהוגדר בפעולה הראשית `main` במחלקה `Driver`.

כתבו את כל המשתנים שאפשר לשים במקום XXX כדי שלאחר ביצוע הפקודה יודפס הפלט `"IntCheck false"`.

שאלה 8

(4 נק') א. הפעולה **First** שלפניכם מקבלת הפניה ch לחוליה הראשונה של שרשרת החוליות של מספרים שלמים ומספר שלם x.

ומספר שלם x.

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

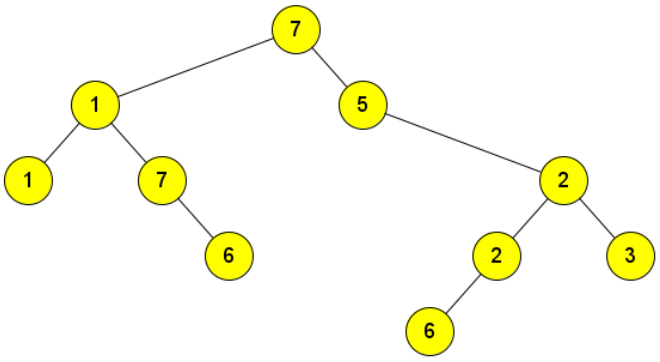
1. נתונה שרשרת החוליות: **ch ->8➔7➔9➔15➔4**
   עקבו אחרי זימון הפעולה ch = First(ch, 3) ורשמו איך תיראה השרשרת אחרי ביצוע הפעולה.
2. מה מבצעת הפעולה First באופן כללי?

(4 נק') ב. כתבו את הפעולה First בצורה הלא רקורסיבית.

(10 נק') ג. הפעולה **Second** שלפניכם מקבלת הפניה לשורש של עץ בינארי ומספר שלם וחיובי k.

```
public static Node<int> Second(BinNode<int> bt, int k){
   Node<int> list = Third(bt, null, k);
   return list;
 }
public static Node<int> Third(BinNode<int> bt, Node<int> list,
                              int k) {
   if(bt!=null)
    {
     if(k>0)
      {
        list = Third(bt.GetLeft(), list, k-1);
        list = Third(bt.GetRight(), list, k-1);
      }
     else
        list = First(list, bt.GetValue());
    }
    return list;
 }
```

1. עקבו אחרי זימון הפעולה Second(bt,3) ורשמו מה תחזיר הפעולה עבור העץ שלפניכם:



2. מה מבצעת הפעולה Second באופן כללי?

<u>שאלה 9</u>

מערכת הניהול של תהליך הרישום לכנס מדעי כוללת את שתי המחלקות הבאות: המחלקה "מושב" (Session) והמחלקה "ניהול כנס" (ConferenceManager).

למחלקה "מושב" (Session) התכונות הבאות:

- שם המושב – nameSession
- תור שמות המשתתפים – participants
- קיבולת מקסימלית – maxCapacity
- מספר המשתתפים בפועל – numParticipants
- סימן אם המושב הוא "קריטי" – isCritical. המושב הוא קריטי, אם נשארו פחות מחמישה מקומות פנויים.

למחלקה "ניהול כנס" ConferenceManager שתי תכונות: שם הכנס ורשימת מושבים:

```
public class ConferenceManager {
   private string nameConference;
   private Node<Session> sessions;
}
```

אפשר להניח שבשתי המחלקות הוגדרו פעולות Get לכל התכונות.

(3 נק') א.

כתבו במחלקה Session פעולה בונה המקבלת את שם המושב ואת הקיבולת המקסימלית. הפעולה מאתחלת מושב ללא משתתפים:

```
public Session (string nameSession, int maxCapacity)
```

(7 נק') ב.

ניתן להוסיף משתתף למושב **רק** אם יש מקום פנוי והמשתתף עדיין לא קיים ברשימת המשתתפים הרשומים במושב זה.

כתבו במחלקה Session פעולה המקבלת שם משתתף ומנסה לרשום אותו למושב. אם הרישום אפשרי, הפעולה תוסיף את שם המשתתף לסוף התור, תעדכן את התכונות הרלוונטיות ותחזיר ערך true. אם הרישום אינו אפשרי, הפעולה תדפיס הודעה מתאימה ותחזיר ערך false.

כותרת הפעולה:

```
public bool RegisterParticipant(string name)
```

(8 נק') ג.

עקב ביקוש רב, הוחלט להחמיר את כללי הרישום למושבים "קריטיים": משתתף יכול להירשם למושב קריטי **רק** אם הוא אינו רשום בשום <u>מושב קריטי</u> אחר.

כתבו במחלקה ConferenceManager פעולה המקבלת שם משתתף ושם מושב. הפעולה תבדוק אם ניתן לרשום את המשתתף למושב. אם כן – הפעולה תבצע רישום ותחזיר ערך true. אם אין מושב רצוי או שהרישום אינו אפשרי, הפעולה תדפיס הודעה מתאימה ותחזיר ערך false.

כותרת הפעולה:

```
public bool TryRegister(string nameSession, string name)
```

חובה להשתמש בפעולה RegisterParticipant.

מבחן ב-JAVA

הנחיות כלליות לנבחנים:

בנספחי הבחינה מובאים ממשקים (interface) הכוללים פונקציות ומחלקות עזר, יש להקפיד ולהשתמש בהן בשאלות הרלוונטיות בבחינה על מנת להימנע מאיבוד ניקוד.

מבחן ב-JAVA

חלק א׳

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

<u>שאלה 1</u>

הגדרה:

סדרת מספרים נקראת ״עולה – יורדת״ אם עד מספר מסוים המספרים הולכים וגדלים וממספר זה ואילך, הם הולכים וקטנים.

לדוגמה, הסדרה שלפניכם היא סדרה ״עולה – יורדת״:

```
4, 9, 13, 20, 45, 33, 18, 1
```

(14 נק׳) א. כתבו פעולה המקבלת מחסנית של מספרים שלמים. הפעולה תבדוק אם הערכים שבמחסנית מהווים סדרה ״עולה – יורדת״. אם כן – הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

4
9
13
20
45
33
18
1

יש לשמור על שלמות המחסנית בסוף הפעולה.

כותרת הפעולה:

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

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

<u>שאלה 2</u>

(6 נק׳) א. כתבו פעולה המקבלת הפניה לחוליה הראשונה בשרשרת חוליות של מספרים שלמים ומספר שלם חיובי `pos`. הפעולה תחזיר איבר שנמצא במקום `pos` בשרשרת. אם אין מקום `pos` בשרשרת, הפעולה תחזיר ערך (-1).

יש להניח כי המיקום נספר החל מ-1, כאשר `pos = 1` מתייחס לחוליה הראשונה.

```
public static int valueAt(Node<Integer> ch, int pos)
```

(8 נק׳) ב. כתבו פעולה המקבלת שתי הפניות לחוליות הראשונות של שתי שרשראות של מספרים שלמים `ch1` ו – `ch2`. הפעולה תיצור שרשרת חדשה לפי הכלל הבא:

- ראשית מכניסים את האיבר הראשון מהשרשרת `ch1`, ואחריו – את האיבר האחרון מהשרשרת `ch2`;
- מכניסים את האיבר השני מהשרשרת `ch1`, ואחריו את האיבר שלפני האחרון מהשרשרת `ch2`;
- וכך הלאה.

הפעולה תחזיר הפניה לחוליה הראשונה של השרשרת החדשה.

<u>אפשר להניח שאורכן של שתי השרשראות שווה.</u>

כותרת הפעולה:

```
public static Node<Integer> merge (Node<Integer>ch1,
                                   Node<Integer>ch2)
```

(2 נק׳) ג. מהן הסיבוכיות של הפעולות שכתבתם בסעיפים א׳ ו-ב׳? **הסבירו את תשובתכם.**

שאלה 3

בפרויקט פיתוח רכב אוטונומי הוגדרו שלושה ממשקים ושתי מחלקות:

```
public interface Steering {
    void turnLeft();
    void turnRight(int degrees);
}
public interface Navigation{ void setDestination(String dest); }
public interface Sensing { void activateLidar(); }
public class AutonomousCar implements Steering, Navigation,
                                      Sensing {
    private String modelName;
    private double batteryLevel;
    // ...
}
public class TestDrive {
    public static void main(String[] args) {
        AutonomousCar car1 = new AutonomousCar("Tesla Y", 98.5);
        AutonomousCar car2 = new AutonomousCar("Waymo One");
        // ( *****)
    }
}
```

(4 נק׳) א. השלימו את המחלקה AutonomousCar: כתבו כותרות לפעולות החסרות בה, כולל כל הפעולות הבונות (constructors) הנדרשות כדי שהקוד בפעולה main יהיה תקין, וכן את כל הפעולות הנדרשות למימוש הממשקים. **אין צורך לממש את גוף הפעולות**.

(12 נק׳) ב. בכל אחד מסעיפים הבאים השורה (*****) תוחלף בשורת קוד משורות הקוד 1–6 מהטבלה הבאה. עבור כל אחד מהסעיפים, כתבו אם הקוד תקין או לא. אם הקוד אינו תקין – יש להסביר למה הוא אינו תקין, יש לציין את סוג השגיאה (קומפילציה או זמן ריצה).

שימו לב: אין קשר בין סעיפים.

1	`Steering s1 = car1;` `s1.turnRight(45);`
2	`Navigation n2 = new Navigation();` `n2.setDestination("Tel Aviv");`
3	`Navigation n3 = new AutonomousCar("Audi A8", 60.0);` `Steering s3 = (AutonomousCar)n3;`
4	`Navigation n4 = new AutonomousCar("Mercedes");` `n4.activateLidar();`

5	`Steering s5 = new AutonomousCar("Ford");`
	`Sensing sn5 = (Sensing) s5 ;`
	`sn5.activateLidar();`

6	`Sensing sn6 = new AutonomousCar("Nissan", 70.0);`
	`((AutonomousCar) sn6).turnLeft();`
	`((Navigation) sn6).setDestination("Haifa");`

חלק ב'

ענו על <u>שתיים</u> מבין השאלות 4–6 (ערך כל שאלה – 16 נקודות).

<u>שאלה 4</u>

(6 נק') א.

הגדרה:

שני מספרים שלמים וחיוביים num1 ו-num2 נקראים "זרים זה לזה" אם אין להם שום ספרה משותפת.

לדוגמה:

המספרים 81224 ו-973656 הם זרים, וגם המספרים 1000 ו-289 הם זרים.

המספרים 14 ו-21 הם לא זרים כי יש להם ספרה משותפת, 1.

כתבו פעולה המקבלת שני מספרים שלמים חיוביים. הפעולה תבדוק אם הם "זרים". אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

כותרת הפעולה:

```
public static boolean areStrangers(int num1, int num2)
```

(7 נק') ב.

כתבו פעולה המקבלת תור של מספרים שלמים חיוביים. הפעולה תבדוק אם כל הערכים בתור זרים זה לזה. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

כותרת הפעולה:

```
public static boolean isStrangersQueue(Queue<Integer> q)
```

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

שאלה 5

חברת סטארט-אפ פיתחה פלטפורמה לשירותי רשת. הטבלה שלפניכם מייצגת את היררכיית היישומים (Applications) במערכת.

מחלקה	יורשת מ-	תיאור
Application	Object	מחלקת אב – יישום כללי במערכת
WebService	Application	שירות רשת (Web)
MobileApp	Application	אפליקציה לנייד
BackendService	WebService	שירות צד שרת
FrontendApp	WebService	שירות צד לקוח
DataQuery	MobileApp	כלי לשליפת נתונים באפליקציה ניידת

(2 נק׳) א. שרטטו את עץ הירושה הכולל של המחלקות (Class Diagrams).

(8 נק׳) ב. נתון קטע קוד מתוך הפעולה הראשית (main):

```
Application a1 = new Application();
Application a2 = new FrontendApp();
Application a3 = new MobileApp();
WebService ws1 = new BackendService();
MobileApp ma1 = new DataQuery();
```

לפניכם שמונה קטעי קוד. עבור כל אחד מהסעיפים יש לציין אם הקוד תקין או לא. אם הקוד אינו תקין – יש להסביר למה הוא אינו תקין ולציין את סוג השגיאה (שגיאת קומפילציה/הידור או שגיאת זמן ריצה).

שימו לב: אין קשר בין הקטעים.

1	`WebService ws2 = new Application();`
2	`WebService ws3 = a2;`
3	`WebService ws4 = (WebService)a2;`
4	`WebService ws5 = (WebService)a3;`
5	`BackendService bs1 = (BackendService)ws1;`
6	`MobileApp ma2 = (MobileApp)a1;`
7	`Application a4 = (MobileApp)(new DataQuery());`
8	`FrontendApp fa1 = (WebService)(new BackendService());`

(6 נק׳) ג. לכל שירות במערכת יש עלות תפעול משלו (Cost). המטרה היא לחשב את סכום העלויות של כל היישומים שהם שירותי הרשת (WebService).

1. באילו מהמחלקות יש להגדיר את הפעולה `public double getCost()` המחזירה את עלות התפעול? **הסבירו את תשובתכם**.
2. כתבו פעולה סטטית בשם `sumWebServiceCost` המקבלת מערך של יישומים ומחזירה את סכום העלויות של כל **שירותי הרשת**. כותרת הפעולה:

```
public static double sumWebServiceCost(Application[] arr)
```

הניחו שהפעולה `getCost()` מומשה כראוי.

<u>שאלה 6</u>

נתונה המחלקה SpecNode:

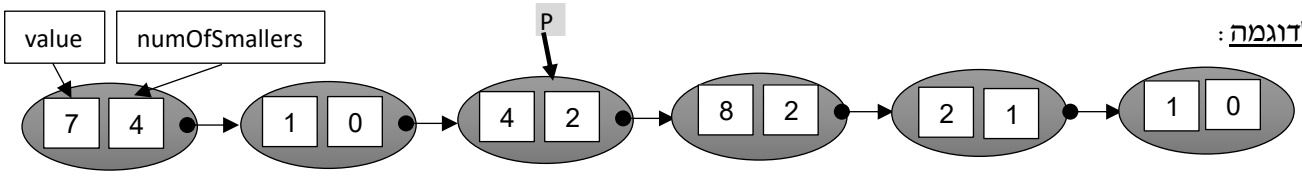
```
public class SpecNode {
    private int value;
    private int numOfSmallers;
    private SpecNode next;

   public SpecNode (int val) {
      this.value = val;
      this. numOfSmallers = 0;
      this.next = null;
    }
}
```

לכל תכונה במחלקה הוגדרו פעולות get/set.

בשרשרת חוליות מסוג SpecNode:

- value – מספר שלם.
- numOfSmallers – מספר החוליות המופיעות **אחרי** החוליה הנוכחית בשרשרת שערך ה-value שלהן **קטן** מערך ה-value של החוליה הנוכחית.
- next – מצביע לחוליה הבאה.

<u>לדוגמה</u>:



- ערך ה-value של החוליה הראשונה הוא 7, ואחריה יש ארבע חוליות שערכי ה-value שלהן קטן מ-7 ובגלל זה ערך התכונה numOfSmallers של החוליה הראשונה הוא 4.
- P מצביע על חוליה עם value=4, והתכונה numOfSmallers=2 מציינת כי יש שתי חוליות אחריה שערכי ה-value שלהן קטנים מ-4 (כלומר 1 ו-2).

(14 נק׳) א.

כתבו פעולה חיצונית המכניסה חוליה חדשה לשרשרת במיקום pos ועם ערך val. הפעולה מקבלת הפניה לחוליה הראשונה של שרשרת חוליות מסוג SpecNode, מספר שלם val ומספר שלם חיובי pos. (המיקום הראשון בשרשרת הוא 1). **אפשר להניח כי מיקום pos חוקי (אינו עולה על אורך רשימה+1)**.

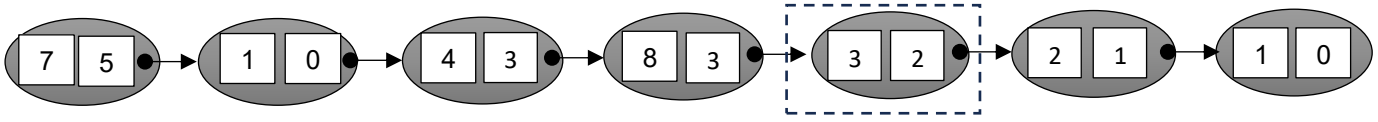
לאחר ההוספה, יש לעדכן את כל ערכי numOfSmallers בהתאם.

כותרת הפעולה:

```
public static SpecNode addNum(SpecNode list, int val, int pos)
```

<u>לדוגמה</u>:

אחרי זימון הפעולה addNum(list, 3, 5) השרשרת שתתקבל תהיה:



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

חלק ג'

ענו על <u>שתיים</u> מבין השאלות 7–9 (ערך שאלה – 18 נקודות).

<u>שאלה 7</u>

נתון פרויקט שהוגדרו בו שלוש מחלקות `Employee, Manager, Intern` ומחלקה `Driver`, כך:

```
public class Employee {
    private int seniority; // ותק

    public Employee(int seniority) {
        this.seniority = seniority;
    }
    public int getSeniority() {
        return this.seniority;
    }
    public boolean check(Employee e) {
        System.out.println("EmpCheck");
        return this.seniority > e.seniority;
    }
}
```

```
public class Manager extends Employee {
    public Manager(int seniority) {
        super(seniority);
    }
    public boolean check(Manager m) {
        System.out.println("MgrCheck");
        return this.getSeniority() == m.getSeniority();
    }
}
```

```
public class Intern extends Employee {
    private int mentorId;

    public Intern(int seniority, int mentorId) {
        super(seniority);
        this.mentorId = mentorId;
    }
    public boolean check(Employee e) {
        System.out.println("IntCheck");
        if (!(e instanceof Intern)) {
            return true;
        }
        Intern other = (Intern) e;
        return this.mentorId == other.mentorId;
    }
}
```

```
public class Driver
```

```
{
    public static void main(String[] args)
    {
        Employee e1 = new Manager(15);
        Manager  m1 = new Manager(15);
        Employee e2 = new Employee(10);
        Employee i1 = new Intern(2, 800);
        Intern   i2 = new Intern(2, 900);
        Intern   i3 = new Intern(5, 800);


(1)     System.out.println(e1.check(m1));
(2)     System.out.println(m1.check(e1));
(3)     System.out.println(m1.check(m1));
(4)     System.out.println(i1.check(e2));
(5)     System.out.println(i1.check(i2));
(6)     System.out.println(i1.check(i3));
(7)     System.out.println(e2.check(i1));

    }
}
```

(14 נק׳) א. עבור כל אחת מהפקודות 1–7 כתבו מה יהיה הפלט של הפקודה.

(4 נק׳) ב. נתונה הפקודה: `System.out.println(i1.check(`**XXX**`)` כאשר **XXX** הוא משתנה שהוגדר בפעולה הראשית `main` במחלקה `Driver`.

כתבו את כל המשתנים שאפשר לשים במקום XXX כדי שלאחר ביצוע הפקודה יודפס הפלט `"IntCheck false"`.

שאלה 8

(4 נק') א. הפעולה **first** שלפניכם מקבלת הפניה ch לחוליה הראשונה של שרשרת החוליות של מספרים שלמים ומספר שלם x.

```
public static Node<Integer> first(Node<Integer> ch, int x){
    if(ch == null)
        return new Node<Integer>(x);
    ch.setNext(first(ch.getNext(),x));
    return ch;
}
```

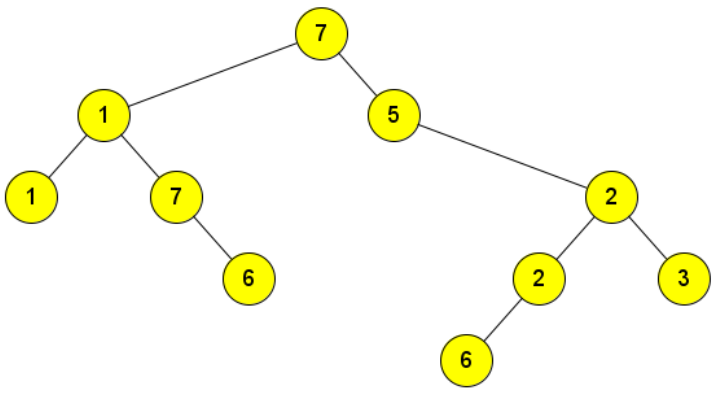
1. נתונה שרשרת החוליות: **ch ->8→7→9→15→4**
עקבו אחרי זימון הפעולה ch = first(ch, 3) ורשמו איך תיראה השרשרת אחרי ביצוע הפעולה.
2. מה מבצעת הפעולה first באופן כללי?

(4 נק') ב. כתבו את הפעולה first בצורה הלא רקורסיבית.

(10 נק') ג. הפעולה **second** שלפניכם מקבלת הפניה לשורש של עץ בינארי ומספר שלם וחיובי k.

```
public static Node<Integer> second(BinNode<Integer> bt, int k){
   Node<Integer> list = third(bt, null, k);
   return list;
 }
public static Node<Integer> third(BinNode<Integer> bt,
                                  Node<Integer> list, int k) {
   if(bt!=null)
    {
     if(k>0)
      {
        list = third(bt.getLeft(), list, k-1);
        list = third(bt.getRight(), list, k-1);
      }
     else
        list = first(list, bt.getValue());
    }
    return list;
 }
```

1. עקבו אחרי זימון הפעולה second(bt,3) ורשמו מה תחזיר הפעולה עבור העץ שלפניכם:



2. מה מבצעת הפעולה second באופן כללי?

<u>שאלה 9</u>

מערכת הניהול של תהליך הרישום לכנס מדעי כוללת את שתי המחלקות הבאות: המחלקה "מושב" (Session) והמחלקה "ניהול כנס" (ConferenceManager).

למחלקה "מושב" (Session) התכונות הבאות:

- שם המושב – nameSession
- תור שמות המשתתפים – participants
- קיבולת מקסימלית – maxCapacity
- מספר המשתתפים בפועל – numParticipants
- סימן אם המושב הוא "קריטי" – isCritical. המושב הוא קריטי, אם נשארו פחות מחמישה מקומות פנויים.

למחלקה "ניהול כנס" ConferenceManager שתי תכונות: שם הכנס ורשימת מושבים:

```
public class ConferenceManager {
   private String nameConference;
   private Node<Session> sessions;
}
```

אפשר להניח שבשתי המחלקות הוגדרו פעולות get לכל התכונות.

(3 נק') א.

כתבו במחלקה Session פעולה בונה המקבלת את שם המושב ואת הקיבולת המקסימלית. הפעולה מאתחלת מושב ללא משתתפים:

```
public Session (String nameSession, int maxCapacity)
```

(7 נק') ב.

ניתן להוסיף משתתף למושב **רק** אם יש מקום פנוי והמשתתף עדיין לא קיים ברשימת המשתתפים הרשומים במושב זה.

כתבו במחלקה Session פעולה המקבלת שם משתתף ומנסה לרשום אותו למושב. אם הרישום אפשרי, הפעולה תוסיף את שם המשתתף לסוף התור, תעדכן את התכונות הרלוונטיות ותחזיר ערך true. אם הרישום אינו אפשרי, הפעולה תדפיס הודעה מתאימה ותחזיר ערך false.

כותרת הפעולה:

```
public boolean registerParticipant(String name)
```

(8 נק') ג.

עקב ביקוש רב, הוחלט להחמיר את כללי הרישום למושבים "קריטיים": משתתף יכול להירשם למושב קריטי **רק** אם הוא אינו רשום בשום <u>מושב קריטי</u> אחר.

כתבו במחלקה ConferenceManager פעולה המקבלת שם משתתף ושם מושב. הפעולה תבדוק אם ניתן לרשום את המשתתף למושב. אם כן – הפעולה תבצע רישום ותחזיר ערך true. אם אין מושב רצוי או שהרישום אינו אפשרי, הפעולה תדפיס הודעה מתאימה ותחזיר ערך false.

כותרת הפעולה:

```
public boolean tryRegister(String nameSession, String name)
```

חובה להשתמש בפעולה registerParticipant.

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, ואם לא – הפעולה תחזיר "שקר".	**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 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): $[x_1, x_2, ..., x_n]$	**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, ואם לא – הפעולה תחזיר "שקר".	**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, ואם לא – הפעולה תחזיר "שקר".	**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.

Queue<T>()	הפעולה בונה תור ריק.
bool IsEmpty()	הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ואם לא – הפעולה תחזיר "שקר".
void Insert (**T** x)	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): $[x_1, x_2, ..., x_n]$

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 חוליה בינרית BINNODE<T>

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

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

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