השאלות

בשאלון זה שני פרקים. יש לענות על שאלות משני הפרקים, לפי ההוראות בכל פרק.

הערה: בכל שאלה שנדרשת בה קליטה, אין צורך לבדוק את תקינות הקלט.

לפותרים בשפת Java : בכל שאלה שנדרשת בה קליטה, הניחו שבתוכנית כתובה ההוראה:

```
Scanner input = new Scanner (System.in);
```

> **שימו לב:** בכל שאלה שנדרש בה מימוש אפשר להשתמש בפעולות של המחלקות: תור, מחסנית, עץ בינרי וחוליה, בלי לממש אותן. אם משתמשים בפעולות נוספות, יש לממש אותן.

פרק ראשון (50 נקודות)

ענו על שתיים מן השאלות 1–3 (לכל שאלה – 25 נקודות).

1. נתונה המחלקה **Order** – הזמנה של לקוח, ולה שתי תכונות:

- id – מספר זהות של הלקוח, מטיפוס שלם
- count – כמות המוצרים שהוזמנו, מטיפוס שלם

הניחו שיש פעולות get/Get ו־ set/Set לתכונות המחלקה, ופעולה בונה המקבלת ערכים בעבור תכונות המחלקה.

בחברת המשלוחים "ברק" נבנה תור – qOrder מטיפוס **Order** , השומר את ההזמנות השונות של הלקוחות ביום מסוים, שחברת המשלוחים צריכה לספק.

הערות:

- ייתכן שיהיו בתור כמה הזמנות של אותו הלקוח – id (אם יש ללקוח יותר מהזמנה אחת באותו היום).
- מיקום ההזמנות בתור אינו לפי סדר כלשהו (וגם ההזמנות של אותו הלקוח יכולות להופיע במקומות שונים בתור).

בסוף היום, כדי לייעל את המשלוחים, מבצעים בתור חדש איחוד הזמנות לפי מזהה לקוח (id), כך שלכל לקוח נשמרת בתור הזמנה אחת בלבד, עם סך כל המוצרים שהוא הזמין בכל ההזמנות (count). כך למשל, אם יש בתור qOrder 3 הזמנות של אותו הלקוח: הזמנה של 20 מוצרים, הזמנה של 15 מוצרים והזמנה של 30 מוצרים, לאחר האיחוד יופיע הלקוח רק פעם אחת בתור החדש – עם הזמנה של 65 מוצרים.

א. **(1)** ממשו את הפעולה שלפניכם:

Java – public static Queue<Order> uniteOrders (Queue<Order> qOrder)

C# – public static Queue<Order> UniteOrders (Queue<Order> qOrder)

הפעולה מקבלת תור שיש בו הזמנות – qOrder ובו אותו לקוח יכול להופיע יותר מפעם אחת, ומחזירה תור חדש ובו יש איחוד הזמנות כפי שהוסבר לעיל.

הערות:

- אין חשיבות לסדר ההזמנות לאחר איחוד ההזמנות.
- אין להשתמש בסעיף זה במערך או ברשימה מקושרת (או בשום מבנה נתונים אחר פרט לתור).
  פתרון הכולל שימוש בהם לא יזוכה בנקודות.
- אפשר לשנות את התור שהתקבל.

(2) מהי סיבוכיות זמן הריצה של הפעולה? נמקו את תשובתכם.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד 3

ב. **(1)** "לקוח רגיל" הוא לקוח שהזמין פחות מ- 10 מוצרים **סך הכול** ביום מסוים, ו"לקוח מועדף" הוא לקוח שהזמין 10 מוצרים ומעלה **סך הכול** ביום מסוים.

ממשו את הפעולה שלפניכם:

Java – public static Queue<Integer> preferredClients (Queue<Order> qOrder)

C# – public static Queue<int> PreferredClients (Queue<Order> qOrder)

הפעולה מקבלת תור הזמנות של יום מסוים – qOrder מטיפוס **Order** (תור "לא מאוחד", שבו אותו לקוח יכול להופיע כמה פעמים), ומחזירה תור מטיפוס **שלם**. בתחילת התור המוחזר יופיעו מספרי הזהות (id) של כל מי שהוא "לקוח מועדף", ואחריהם יופיעו מספרי הזהות (id) של כל מי שהוא "לקוח רגיל" (ללא חשיבות לסדר הלקוחות בכל קבוצה).

אפשר להשתמש בפעולה שכתבתם בסעיף א.

<u>הערות</u>:

- כל לקוח (id) יופיע פעם אחת בלבד בתור המוחזר.
- <u>אין להשתמש בסעיף זה במערך או ברשימה מקושרת</u> (או בשום מבנה נתונים אחר פרט לתור). פתרון הכולל שימוש בהם לא יזוכה בנקודות.
- אפשר לשנות את התור שהתקבל.

(2) מהי סיבוכיות זמן הריצה של הפעולה? נמקו את תשובתכם.

/המשך בעמוד 4/

2. נתונות שתי הפניות לשרשרות של חוליות:

numHead – הפניה לשרשרת חוליות של מספרים מטיפוס שלם.

charHead – הפניה לשרשרת חוליות מטיפוס תו (char), שבה מיוצגות פעולות החשבון חיבור וחיסור ('+', '−').

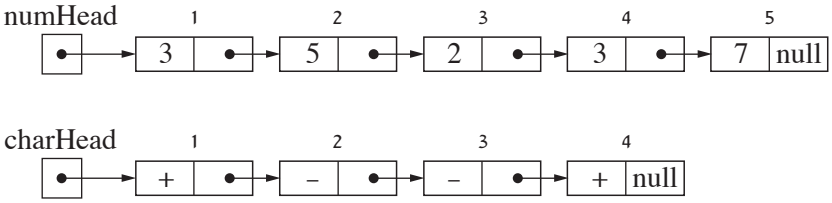
הערה: כמות החוליות בשרשרת המספרים גדולה ב־ 1 מכמות החוליות בשרשרת התווים.

שילוב בין שתי השרשרות מייצג סדרת פעולות חשבון, כמפורט לפניכם:

התו במיקום 1 בשרשרת התווים מתאר את הפעולה שבאה לפני המספר שבמיקום 2 בשרשרת המספרים, התו במיקום 2 מתאר את הפעולה שבאה לפני המספר שבמיקום 3, התו במיקום 3 מתאר את הפעולה שבאה לפני המספר שבמיקום 4, וכן הלאה עד סוף שתי השרשרות.

שימו לב: אין פעולת חשבון לפני המספר הראשון בשרשרת המספרים.

דוגמה: שתי השרשרות שלפניכם מייצגות את סדרת פעולות החשבון: $3+5-2-3+7$.



א. ממשו את הפעולה שלפניכם:

Java – public static int eval (Node<Integer> numHead, Node<Character> charHead, int len)

C# – public static int Eval (Node<int> numHead, Node<char> charHead, int len)

len הוא כמות **המספרים** שנרצה לחשב מתוך שרשרת המספרים (אין פעולת חשבון לפני המספר הראשון ולכן כמות התווים היא len–1 בהתאם).

הפעולה תחזיר את תוצאת החישוב של len מספרים ברצף, החל מתחילת שרשרת המספרים ותחילת שרשרת התווים.

הניחו ש־ len גדול מ־ 0 . אין לשנות את השרשרות שהתקבלו.

הערה: אם len גדול מכמות החוליות בשרשרת המספרים, הפעולה תחזיר את תוצאת החישוב של כל המספרים בשרשרת.

דוגמה: בעבור שתי השרשרות בדוגמה שלעיל ו־ $\text{len} = \mathbf{1}$, הפעולה תחזיר 3 , כיוון שהוא המספר הראשון בשרשרת המספרים (היות שהוא המספר הראשון והיחיד, אין שום פעולת חשבון שצריך לבצע).

דוגמה נוספת: בעבור אותן שתי שרשרות ו־ $\text{len} = \mathbf{3}$, הפעולה תחזיר 6 ($3+5-2=6$).

דוגמה נוספת: בעבור אותן שתי שרשרות ו־ $\text{len} = \mathbf{8}$, הפעולה תחזיר 10 ($3+5-2-3+7=10$).

הסבר: מספר החוליות בשרשרת המספרים קטן מ־ 8 , ולכן הפעולה מחזירה את תוצאת החישוב של כל המספרים בשרשרת.

(שימו לב: המשך השאלה בעמוד הבא.)

ב. ממשו את הפעולה שלפניכם:

Java – public static boolean match (Node<Integer> numHead, Node<Character> charHead, int len, int val)

C# – public static bool Match (Node<int> numHead, Node<char> charHead, int len, int val)

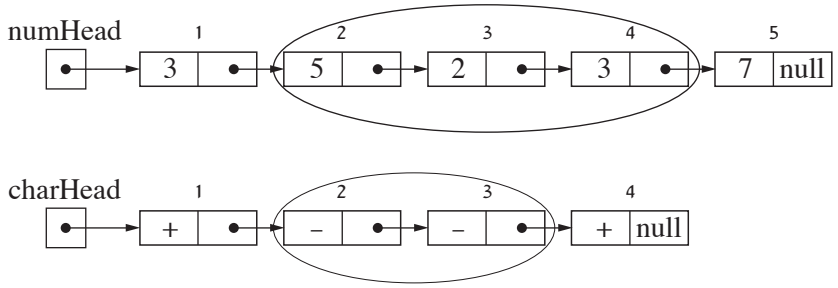
הפעולה תחזיר true אם קיימת חוליה **במיקום כלשהו** בשרשרת המספרים וחוליה **במיקום זהה** בשרשרת התווים, שמהם והלאה יש **len** מספרים ברצף שתוצאת החישוב שלהם היא **val** (כלומר אם מתחילים את החישוב **מן החוליה השנייה** בשרשרת המספרים, גם פעולת החשבון הראשונה תהיה **החוליה השנייה** בשרשרת התווים). אחרת הפעולה תחזיר false .

אפשר להשתמש בפעולה שכתבתם בסעיף א. הניחו ש־ len גדול מ־ 0 . אין לשנות את השרשרות שהתקבלו.

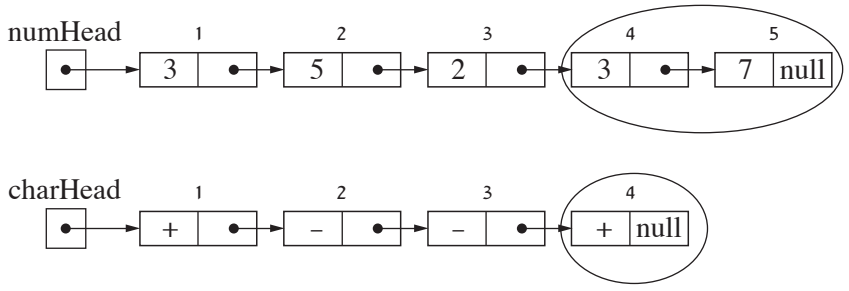
שימו לב: אין פעולת חשבון לפני המספר הראשון בחישוב (מספר התווים הנדרש הוא len–1).

הערה: אם מחוליה כלשהי יש פחות מ־ len מספרים עד סוף השרשרת – החישוב מתבצע עד סוף השרשרת.

דוגמה: בעבור שתי השרשרות המוצגות בדוגמה שלעיל, $len = 3$ ו־ $val = 0$, הפעולה תחזיר **true** , כי החל **מן החוליה השנייה** בשרשרת המספרים ו**מן החוליה השנייה** בשרשרת התווים יש 3 מספרים ברצף שתוצאת החישוב שלהם היא 0 ($0 = 3 - 2 - 5$).



דוגמה נוספת: בעבור שתי השרשרות המוצגות בדוגמה שלעיל, $len = 3$ ו־ $val = 10$, הפעולה תחזיר **true** , כי החל **מן החוליה הרביעית** בשרשרת המספרים ו**מן החוליה הרביעית** בשרשרת התווים ועד סוף שרשרת המספרים תוצאת החישוב היא 10 ($10 = 7 + 3$).



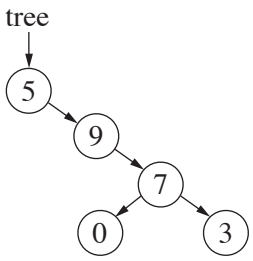
/המשך בעמוד 6/

לפותרים בשפת C#

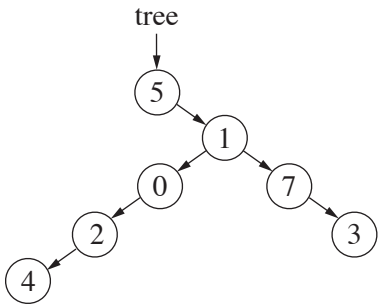
א. נתונה הפעולה Foo שלפניכם:

```
public static int Foo (BinNode<int> tree)
{
    if (tree.GetLeft() == null && tree.GetRight() == null)
        return 0;
    if (tree.GetLeft() == null)
        return Foo (tree.GetRight()) + 1;
    if (tree.GetRight() == null)
        return Foo (tree.GetLeft()) – 1;
    return Foo (tree.GetLeft()) + Foo (tree.GetRight());
}
```

(1) בצעו מעקב אחרי הפעולה Foo והעץ שלפניכם, וכתבו מה הפעולה מחזירה. יש להציג את המעקב.



(2) בצעו מעקב אחרי הפעולה Foo והעץ שלפניכם, וכתבו מה הפעולה מחזירה. יש להציג את המעקב.



(3) בעבור עץ שיש לו 6 צמתים, מהו המספר הכי גבוה שהפעולה Foo יכולה להחזיר? הציגו את העץ.

(4) בעבור עץ שיש לו 6 צמתים, מהו המספר הכי נמוך שהפעולה Foo יכולה להחזיר? הציגו את העץ.

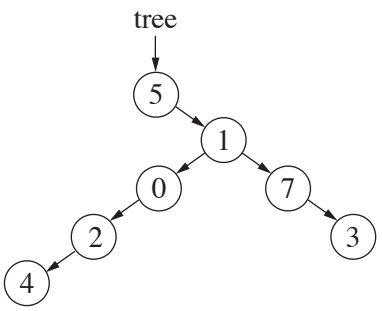
(5) כתבו בקצרה מה הפעולה Foo מחזירה בעבור עץ כלשהו שאינו ריק.

(שימו לב: המשך השאלה בעמוד הבא.)

ב. נתונה הפעולה Goo שלפניכם:

```
public static bool Goo (BinNode<int> tree)
{
    if (tree == null)
        return true;
    if (Foo (tree) != 0)
        return false;
    return Goo (tree.GetLeft()) && Goo (tree.GetRight());
}
```

(1) בצעו מעקב אחרי הפעולה Goo והעץ שלפניכם, וכתבו מה הפעולה מחזירה. יש להציג את המעקב.
בסעיף זה אין צורך להציג מעקב אחרי הפעולה Foo .



(2) הציגו עץ שיש לו <u>מעל</u> 3 צמתים שבעבורו הפעולה Goo תחזיר תוצאה <u>הפוכה</u> מן התוצאה שהתקבלה בתת־סעיף ב(1).

/המשך בעמוד 10/

לפותרים בשפת C#

באקדמיה למוזיקה "**שיר חדש**" פותחה מערכת ממוחשבת ובה המחלקות האלה:

Instrum – כלי נגינה, **Bow** – כלי קשת , **Wind** – כלי נשיפה, **Perc** – כלי הקשה, **Musician** – נגן בתזמורת, **Orchestra** – תזמורת.

להלן פירוט תכונות המחלקות:

למחלקה **Instrum** (כלי נגינה) שתי תכונות: name – שם הכלי (מטיפוס מחרוזת), material – החומר שממנו עשוי הכלי (מטיפוס מחרוזת).

למחלקה **Bow** (כלי קשת) שלוש תכונות: name – שם הכלי (מטיפוס מחרוזת), material – החומר שממנו עשוי הכלי (מטיפוס מחרוזת), bowLength – אורך הקשת (מטיפוס שלם).

למחלקה **Wind** (כלי נשיפה) שלוש תכונות: name – שם הכלי (מטיפוס מחרוזת), material – החומר שממנו עשוי הכלי (מטיפוס מחרוזת), numberOfValves – מספר השסתומים בכלי (מטיפוס שלם).

למחלקה **Perc** (כלי הקשה) שלוש תכונות: name – שם הכלי (מטיפוס מחרוזת), material – החומר שממנו עשוי הכלי (מטיפוס מחרוזת), hasSticks – האם יש שימוש במקלות הקשה (מטיפוס בוליאני. אם יש שימוש במקלות – true , ואם לא – false).

למחלקה **Musician** (נגן בתזמורת) שתי תכונות: name – שם הנגן (מטיפוס מחרוזת), instrum – כלי הנגינה שלו (מטיפוס **Instrum**). כל נגן מנגן על כלי אחד בלבד.

למחלקה **Orchestra** (תזמורת) שתי תכונות: arrM – מערך של נגנים (מטיפוס **Musician**), count – כמות הנגנים השמורים במערך (מטיפוס שלם).

הנגנים מופיעים ברצף מתחילת המערך ללא null ביניהם, אך ייתכן שהמערך אינו מלא עד הסוף.

א. **(1)** סרטטו תרשים היירכייה בין כל המחלקות.

יש לסמן ירושה באמצעות החץ ◁—— והכלה באמצעות הסימן ——◆ .

(2) כתבו את כותרות המחלקות ואת התכונות שלהן. הניחו שבכל מחלקה הוגדרו פעולות Get ו־ Set ופעולות בונות.

ב. כתבו במחלקה **Orchestra** פעולה ששמה WithSticks , המחזירה את מספר הנגנים בתזמורת המנגנים בכלי הקשה שיש בהם שימוש במקלות.

ג. מערך arrM "מסודר" הוא מערך שבו מופיעים קודם נגני כלי הקשת, אחר כך נגני כלי הנשיפה, ואחר כך נגני כלי ההקשה.

כתבו במחלקה **Orchestra** פעולה ששמה Insert , המקבלת נגן – musician שאינו מופיע במערך arrM .

הניחו שלפני תחילת הפעולה המערך "מסודר", ושיש בתזמורת לפחות נגן אחד בכל סוג של כלי.

אם אין מקום לנגן נוסף במערך, הפעולה תחזיר false . אחרת, הפעולה תכניס את הנגן למערך במקום המתאים באופן שבו המערך ימשיך להיות "מסודר" (והנגנים יופיעו ברצף בתחילת המערך), ותחזיר true .

הערה: אפשר לשנות את סדר הנגנים, ובלבד שהמערך יהיה "מסודר" בסוף הפעולה.

/המשך בעמוד 20/

לפותרים בשפת C#

לפניכם שלושה סעיפים א–ג שאין קשר ביניהם. ענו על שלושת הסעיפים.

א. לפניכם המחלקה **Father** ללא מימוש הפעולות הבונות, והמחלקה **Son** ללא מימוש הפעולות הבונות ופעולת ToString .

```
public class Father {
    private static int st = 0;
    protected int num;
    public Father() {…}
    public Father (int num) {…}
    public override string ToString() { return "st=" + st + ", num=" + num; }
}
public class Son : Father {
    private char ch;
    public Son (int num, char ch) ...{…}
    public Son (int num) ...{…}
    public override string ToString() {...}
}
```

לפניכם קטע קוד והפלט שהתקבל בעקבות הרצת קטע הקוד:

הקוד		הפלט	
Father x1 = new Father();			
Console.WriteLine (x1);	//1	st=1, num=9	//1
Father x2 = new Father (3);			
Console.WriteLine (x2);	//2	st=1, num=3	//2
Father x3 = new Son (5, 'A');			
Console.WriteLine (x3);	//3	st=2, num=5, ch=A	//3
Father x4 = new Son (7);			
Console.WriteLine (x4);	//4	st=2, num=7, ch=Z	//4

ממשו את הפעולות הבונות במחלקה Father ובמחלקה Son , ואת הפעולה ToString במחלקה Son , בהתאם לעקרונות תכנות מונחה עצמים, כך שיתקבל הפלט הנתון.

אין להוסיף פעולות נוספות למחלקות. תשובה הכוללת פעולות נוספות במחלקות, לא תזוכה בנקודות.

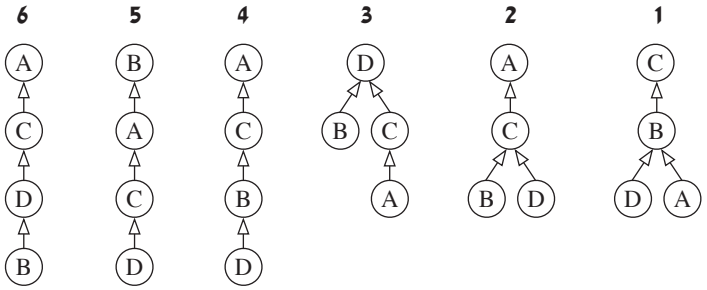
(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד 23

ב. נתונות המחלקות **A**, **B**, **C**, **D** והפעולה Main במחלקה **Tester**, שרצה באופן תקין.

```
public class Tester {
    public static void Main (string[] args) {
        C c1 = new B();
        A a1 = new C();
        A a2 = new B();
        C c2 = new D();
    }
}
```

(1) לפניכם שישה תרשימי היירכייה 1–6 (החץ מסמן יחס ירושה). בנוגע לכל אחד מהם, ציינו אם ייתכן או לא ייתכן שהוא מתאר את יחס ההיירכייה בין המחלקות **A**, **B**, **C**, **D**. אם ציינתם שלא ייתכן, נמקו מדוע.



(2) בסוף הפעולה Main נוספו שורות הקוד שלהלן, שרצות באופן תקין:

```
A a3 = new B();
D d1 = (D)a3;
```

בעקבות זאת, האם תשתנה תשובתכם בתת־סעיף ב(1)? נמקו את תשובתכם.

ג. נתונה המחלקה **BBB**, היורשת ממחלקה **AAA**:

```
public class BBB : AAA {
    private char ch;
    public BBB (int num, char ch) : base (num) {
        this.ch = ch;
    }
    public BBB (char ch) {this.ch = ch;}
    public override int GG (int val) {return base.GG (val) + num;}
    public int MM() {return AAA.c + 2;}
    public bool NN() {return this.MM (10);}
}
```

כתבו במחלקה AAA את **תכונות** המחלקה (כולל הרשאות הגישה) ואת ה**כותרות** של הפעולות הנדרשות כדי שהמחלקה BBB תהיה תקינה, בהתאם לעקרונות תכנות מונחה עצמים.

<u>אין צורך לממש את הפעולות שאת הכותרות שלהן הוספתם במחלקה AAA. אין לשנות את הקוד של המחלקה BBB.</u>

/המשך בעמוד 24/

<u>לפותרים בשפת C#</u>

לפניכם המחלקות **A**, **B**, **C** :

```
public class A {
    public int num;

    public A()
    {
        this.num = 1;
    }
    public A (int num)
    {
        this.num = num;
    }
    public int Func (int val)
    {
        return val * 2;
    }
    public virtual int Func (B other)
    {
        return this.num - other.num;
    }
}
```

```
public class B : A {
    public B() : base (2) { }
    public B(int val) : base (val)
    {
        this.num = this.num + Func (val);
    }
    public int Func (A other)
    {
        return this.num + (other.num * 10);
    }
    public override int Func (B other)
    {
        return this.num + other.num;
    }
}
```

```
public class C : B {
    public C (int val)
    {
        this.num = this.num - val;
    }
    public override int Func (B other)
    {
        return this.num * other.num;
    }
    public int Func (C c)
    {
        return this.num * 4;
    }
}
```

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד 27/

א. לפניכם קטע קוד שרץ ללא שגיאות:

```
A a = new A();
A ab = new B();
A ac = new C(0);
B b = new B(1);
B bc = new C(1);
C c = new C(1);
```

הציגו את העצמים שנוצרו (סוג העצם, סוג ההפניה, וערך התכונה num).

ב. לפניכם המשך קטע הקוד (ממוספר) שרץ ללא שגיאות.

ציינו את מספר שורת ההדפסה, וכתבו מה הקוד מדפיס.

```
1. Console.WriteLine (a.Func (b));
2. Console.WriteLine (a.Func (c));
3. Console.WriteLine (ab.Func ((B)ab));
4. Console.WriteLine (((B)ab).Func (ab));
5. Console.WriteLine (ab.Func (10));
6. Console.WriteLine (a.Func (a.Func (2)));
7. Console.WriteLine (ac.Func (c));
8. Console.WriteLine (c.Func (ac));
9. Console.WriteLine (c.Func (a));
10. Console.WriteLine (((C)bc).Func ((C)bc));
```