

## מדעי המחשב

### הוראות

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה: בשאלון זה שלושה פרקים.

פרק ראשון	-	$(10 \times 1) + (15 \times 1)$	-	25 נקודות
פרק שני	-	$(25 \times 2)$	-	50 נקודות
פרק שלישי	-	$(25 \times 1)$	-	25 נקודות
סך הכול	-		-	100 נקודות

ג. חומר עזר מותר בשימוש: כל חומר עזר, חוץ ממחשבון שיש בו אפשרות תכנות.

ד. הוראות מיוחדות:

(1) רשמו על הכריכה החיצונית של המחברת את שם המסלול שלמדתם.

המסלול הוא אחד משלושת המסלולים האלה: אלגוריתמים, מודלים חישוביים, תכנות מונחה עצמים.

(2) את כל התוכניות שיש לכתוב בשפת מחשב בפרקים הראשון והשני כתבו בשפה אחת בלבד – Java או C#.

הערה: לא יורדו נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.

יש לכתוב במחברת הבחינה בלבד. יש לרשום "טיוטה" בראש כל עמוד המשמש טיוטה.  
כתיבת טיוטה בדפים שאינם במחברת הבחינה עלולה לגרום לפסילת הבחינה.

השאלות בשאלון זה מנוסחות בלשון רבים, אף על פי כן על כל תלמידה וכל תלמיד להשיב עליהן באופן אישי.

**בהצלחה!**

## השאלות

בשאלון זה שלושה פרקים.

יש לענות על שאלות משלושת 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);
```

### פרק ראשון (25 נקודות)

ענו על שאלה 1 – חובה (10 נקודות).

1. "מערך ממוין בחיוביים" הוא מערך מטיפוס שלם שבו כל המספרים הגדולים מ-0 מופיעים בסדר עולה (סדר עולה – מספר הגדול או שווה למספר החיובי הקודם לו).

דוגמה:

המערך arr שלפניכם הוא "מערך ממוין בחיוביים".

	0	1	2	3	4	5	6	7	8
arr	<u>5</u>	<u>9</u>	- 3	<u>17</u>	0	<u>29</u>	- 20	- 40	<u>29</u>

הסבר: המספרים החיוביים במערך (5, 9, 17, 29, 29) מופיעים בסדר עולה.

כתבו פעולה חיצונית ששמה posOrder בשפת Java או PosOrder בשפת C#, המקבלת מערך מטיפוס שלם – arr. הפעולה מחזירה true אם arr הוא "מערך ממוין בחיוביים", אחרת היא מחזירה false. הניחו שיש לפחות שני מספרים חיוביים במערך.

ענו על אחת מן השאלות 2-3 (15 נקודות).

2. "מערך הפרשים" הוא מערך מטיפוס שלם שבו ההפרש בין המספרים בכל שני תאים סמוכים הוא קבוע.

דוגמה למערך הפרשים:

0	1	2	3	4	5
5	9	13	17	21	25

הסבר: ההפרש בין המספרים בכל שני תאים סמוכים במערך זה הוא 4.

ממשו את הפעולה החיצונית שלפניכם:

Java – public static int missingNum (int [] arr)

C# – public static int MissingNum (int [] arr)

הפעולה מקבלת מערך הפרשים – arr שחסר בו תא אחד, ובעקבות זאת יש במערך שני תאים סמוכים שההפרש בין המספרים בהם שונה מן ההפרש הקבוע. הפעולה תחזיר את המספר שאמור להיות בתא החסר.

הניחו שגודל המערך arr שמתקבל הוא לפחות 4.

דוגמה:

עבור המערך arr שלפניכם, הפעולה תחזיר את המספר 10.

	0	1	2	3	4	5
arr	6	<u>8</u>	<u>12</u>	14	16	18

הסבר: בין כל שני תאים סמוכים ההפרש בין המספרים הוא 2, חוץ מן התאים באינדקסים 1 ו-2, שההפרש בין המספרים בהם

הוא 4. זאת מפני שחסר ביניהם תא ובו המספר 10.

3.

בנתיבי תחבורה ציבורית הוצבו מצלמות תנועה, המצלמות כל כלי רכב שעובר על פניהן.

נתונה המחלקה **CarInfo** – מידע על כלי רכב שצולם, ולה שלוש תכונות:

- id – מספר לוחית זיהוי של כלי הרכב שצולם, מטיפוס מחרוזת.
  - privateCar – אם כלי הרכב שצולם הוא פרטי, התכונה היא true, ואם כלי הרכב הוא ציבורי – false.
  - speed – מהירות הנסיעה של כלי הרכב שצולם, מטיפוס שלם.
- הניחו שקיימות פעולות get/Get ו־ set/Set לכל אחת מן התכונות במחלקה.

לכלי רכב שנוסע בנתיב תחבורה ציבורית תירשם עבירת תנועה אם יתקיים לפחות אחד מן המצבים האלה:

– כלי הרכב הוא פרטי.

– כלי הרכב נוסע מעל המהירות המותרת.

א. כתבו פעולה פנימית במחלקה **CarInfo** ששמה illegal בשפת Java או Illegal בשפת C#, המקבלת את מהירות

הנסיעה המותרת – maxSpeed מטיפוס שלם.

הפעולה תחזיר true אם כלי הרכב עבר עבירת תנועה (כלומר, אם כלי הרכב הוא פרטי ו/או נסע מעל המהירות המותרת),

אחרת הפעולה תחזיר false.

נתונה המחלקה **CameraInfo** – מידע של מצלמת תנועה, ולה שלוש תכונות:

- city – קוד העיר שבה המצלמה מוצבת, מטיפוס שלם. הקוד הוא מספר בין 0 ל-99 (כולל).
- יש 100 ערים, ולכל עיר יש קוד ייחודי.

- maxSpeed – המהירות המותרת באזור המצלמה, מטיפוס שלם.

- cars – מערך מטיפוס **CarInfo** של כלי הרכב שצולמו במצלמה (המערך ללא ערכי null).

הניחו שקיימות פעולות get/Get ו־ set/Set לכל אחת מן התכונות במחלקה.

ב. (1) כתבו פעולה פנימית במחלקה **CameraInfo** ששמה allGood בשפת Java או AllGood בשפת C#,

המחזירה true אם כל כלי הרכב שצולמו במצלמה לא עברו עבירת תנועה, אחרת היא מחזירה false.

אפשר להשתמש בפעולה שכתבתם בסעיף א.

(2) כתבו פעולה חיצונית ששמה legalCities בשפת Java או LegalCities בשפת C#.

הפעולה מקבלת מערך – cameras מטיפוס **CameraInfo** של מצלמות שהוצבו ב-100 הערים השונות.

הפעולה תחזיר את כמות הערים שלא אותרה בהן שום עבירת תנועה.

הערה: תיתכן יותר ממצלמה אחת באותה העיר.

אפשר להשתמש בפעולה שכתבתם בסעיף ב(1).

**פרק שני (50 נקודות)**

**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.

ענו על שתיים מן השאלות 4-6 (לכל שאלה – 25 נקודות).

4. בשאלה זו נוספה למחלקה **Queue** הפעולה `size/Size` שלפניכם. אפשר להשתמש בפעולה בלי לממש אותה.

תיאור הפעולה	כותרת הפעולה
הפעולה מחזירה את מספר האיברים בתור.	<b>Java</b> – <code>public int size()</code> <b>C#</b> – <code>public int Size()</code>

ממשו את הפעולה החיצונית שלפניכם:

**Java** – `public static boolean twoSum (Queue<Integer> q, int x)`

**C#** – `public static bool TwoSum (Queue<int> q, int x)`

הפעולה מחזירה `true` אם בתור `q` שהתקבל יש שני מספרים שסכומם שווה לערך הפרמטר `x`. אחרת הפעולה מחזירה `false`.  
דוגמה: עבור התור `q` שלפניכם ו-`x = 10` הפעולה תחזיר `true`, כי יש בתור שני מספרים (9, 1) שסכומם שווה ל-10.

	ראש התור						
q	5	4	<b>1</b>	4	3	15	<b>9</b>

הערות:

- הניחו שבתור `q` יש שני איברים לפחות.
- אין צורך לשמור על התור `q`.
- אין להשתמש בשאלה זו במערך וברשימה מקושרת.

5. נתונה המחלקה **NumCount** – מספר ערכים, ולה שתי תכונות:

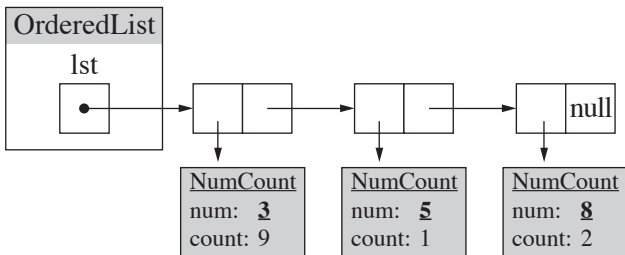
- num – ערך מספרי, מטיפוס שלם.
  - count – מספר המופעים של הערך (num), מטיפוס שלם. המספר גדול או שווה ל-0.
- הניחו שקיימות פעולות get/Set ו- set/Set לכל אחת מן התכונות במחלקה, ופעולה בונה המקבלת ערכים עבור תכונות המחלקה.

נתונה המחלקה **OrderedList** – שרשרת ממוינת, ולה תכונה אחת:

- lst – מצביע על ראש של שרשרת חוליות מטיפוס **NumCount**.

שרשרת החוליות ממוינת לפי סדר עולה של ערך התכונה - num. ערך התכונה num שונה בכל חוליה.

דוגמה: השרשרת שלפניכם מקיימת את תנאי המחלקה (השרשרת ממוינת בסדר עולה לפי ערך התכונה num, וערך התכונה num שונה בכל חוליה).



א. (1) ממשו במחלקה **OrderedList** את הפעולה הפנימית שלפניכם:

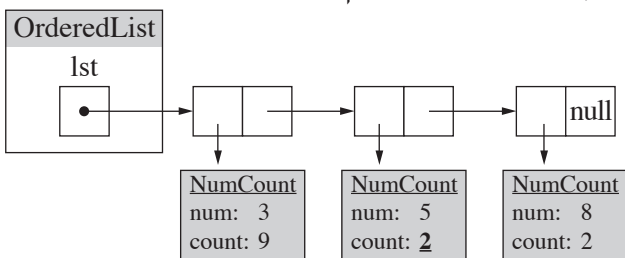
**Java** – public void insertNum (int x)

**C#** – public void InsertNum (int x)

הפעולה מוסיפה את הערך של x לשרשרת באופן שלפניכם:

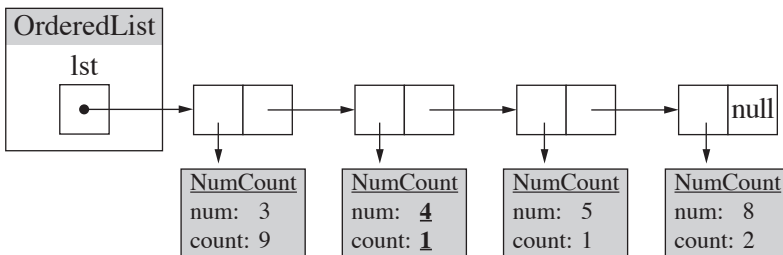
- אם קיימת בשרשרת חוליה שהתכונה num שלה שווה ל- x, הפעולה תגדיל ב-1 את התכונה count (כמות המופעים) באותה החוליה.
- אם השרשרת ריקה או שלא קיימת בשרשרת חוליה שהתכונה num שלה שווה ל- x, הפעולה תכניס חוליה חדשה, שבה התכונה num תהיה שווה ל- x והתכונה count תהיה שווה ל-1, במיקום השומר את הסדר העולה של השרשרת.

דוגמה: עבור השרשרת המוצגת לעיל ו- x = 5, בתום הפעולה תיראה השרשרת כך:



דוגמה נוספת: עבור אותה השרשרת המוצגת לעיל (בדוגמה הראשונה) ו- x = 4, בתום הפעולה תיראה השרשרת כך:

כך:

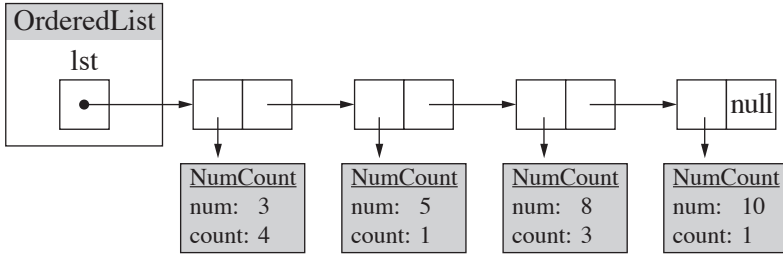


(2) מהי סיבוכיות זמן הריצה של הפעולה שכתבתם בסעיף א(1)? נמקו את תשובתכם.

(שימו לב: המשך השאלה בעמוד הבא.)

ב. "ערך המופע ה- $n$ " הוא הערך שמופיע במקום ה- $n$  לפי הסדר מתחילת השרשרת (בשקלול כמות המופעים – count של כל ערך).

לדוגמה: עבור השרשרת שלפניכם ו- $n = 7$  הפעולה תחזיר את הערך 8.



הסבר: סדר הערכים של השרשרת ברצף, בהתאם לכמות המופעים שלהם, הוא: 3, 3, 3, 3, 5, 8, **8**, 8, 10.

וּבמקום השביעי ברצף מופיע הערך 8. לכן הפעולה תחזיר את הערך 8.  $n = 7$ .

ממשו במחלקה OrderedList את הפעולה הפנימית שלפניכם:

**Java** – public int valueN (int n)

**C#** – public int ValueN (int n)

הפעולה מקבלת את המספר  $n$ , ומחזירה את "ערך המופע ה- $n$ ".

הניחו ש"ערך המופע ה- $n$ " קיים בשרשרת.

6. "מספר ראשוני" הוא מספר המתחלק רק בעצמו וב-1 (גם המספרים 1 ו-2 הם ראשוניים).  
לפניכם הפעולה החיצונית isPrime/IsPrime. אפשר להשתמש בפעולה בלי לממש אותה.

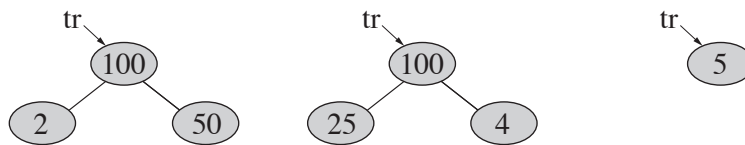
כותרת הפעולה	תיאור הפעולה
Java – public static boolean isPrime (int num)	הפעולה מחזירה true אם הערך num שהתקבל
C# – public static bool IsPrime (int num)	הוא מספר ראשוני, אחרת היא מחזירה false.

א. ממשו את הפעולה החיצונית שלפניכם:

Java – public static boolean addNodes (BinNode<Integer> tr)

C# – public static bool AddNodes (BinNode<int> tr)

הפעולה מקבלת צומת ללא בנים (עלה) שערכו גדול מ-0. אם ערך הצומת הוא מספר ראשוני, הפעולה מחזירה false. אחרת, הפעולה מוסיפה לצומת שני בנים שערך המכפלה שלהם שווה לערך הצומת, והערך של כל אחד מהם גדול מ-1. לאחר ההוספה הפעולה מחזירה true.  
דוגמאות: בתום הפעולה הצמתים יכולים להיראות כך (עבור הערכים 5, 100, 100):



ב. נתונה הפעולה what/What שלפניכם, המקבלת צומת ללא בנים שערכו גדול מ-0.

בשפת C#	בשפת Java
<pre>public static void What (BinNode&lt;int&gt; tr) {     if (AddNodes (tr))     {         What (tr.GetLeft());         What (tr.GetRight());     } }</pre>	<pre>public static void what (BinNode&lt;Integer&gt; tr) {     if (addNodes (tr))     {         what (tr.getLeft());         what (tr.getRight());     } }</pre>

- (1) סרטטו את העץ כפי שהוא ייראה בתום הפעולה what/What עבור צומת ללא בנים – tr שערכו 150. יש להראות מעקב.
- (2) הסבירו מה מבצעת הפעולה what/What.



**פרק שלישי (25 נקודות)**

בפרק זה שאלות בשלושה מסלולים:

אלגוריתמים, עמודים 9-10.

מודלים חישוביים, עמודים 11-12.

תכנות מונחה עצמים בשפת Java, עמודים 14-17; תכנות מונחה עצמים בשפת C#, עמודים 18-21.

יש לענות על שאלה אחת במסלול שלמדתם.

**אלגוריתמים**

אם למדתם מסלול זה, ענו על אחת מן השאלות 7-8 (25 נקודות).

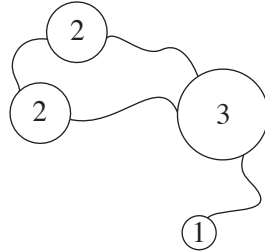
7. בשאלה זו שני סעיפים, א-ב, שאין קשר ביניהם. ענו על שני הסעיפים.

א. לפניכם חמש טענות. בחרו בארבע מהן.

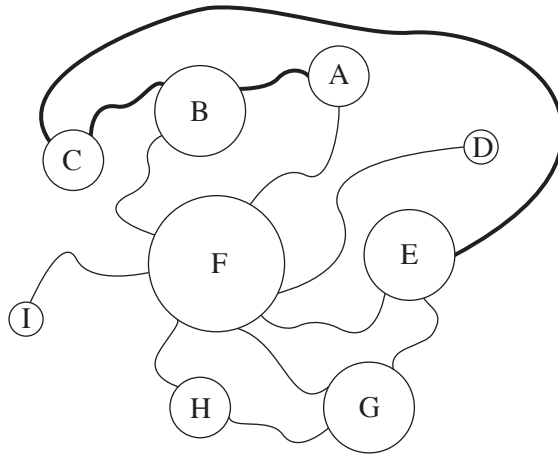
עבור כל אחת מן הטענות שבחרתם, כתבו את מספר הטענה במחברתכם, וציינו אם היא נכונה או לא נכונה. אם ציינתם שהטענה נכונה – נמקו מדוע, ואם ציינתם שהטענה אינה נכונה, הביאו דוגמה נגדית או נמקו מדוע.

1. לכל עץ פורש DFS של אותו גרף  $G$  לא מכונן יש תמיד אותו מספר עלים.
  2. גובה עץ פורש BFS של גרף  $G$  לא מכונן, תמיד קטן או שווה לגובה עץ פורש DFS של אותו הגרף.
  3. גרף  $G$  לא מכונן בעל  $n$  צמתים ו- $2n$  קשתות הוא תמיד לא קשיר.
  4. בכל גרף  $G$  מכונן שבו  $n$  צמתים ו- $m$  קשתות, ו- $n \geq m$ , קיים מעגל.
  5. גרף  $G$  לא מכונן שבו הדרגות של כל הקודקודים גדולות מ-0 הוא תמיד קשיר.
- ב. "עץ פורש מקסימלי" הוא עץ פורש של גרף  $G$  לא מכונן שבו סכום ערכי הקשתות הוא מקסימלי. כתבו אלגוריתם המוצא בגרף  $G$  "עץ פורש מקסימלי".

8. ב"רשת רחובות" המעבר מרחוב לרחוב הוא תמיד דרך כיכר. גודלו (במטרים) של רדיוס של כיכר הוא כמספר הרחובות המחברים אל הכיכר. לדוגמה, הרדיוס של כיכר המחברת 3 רחובות הוא 3 מטרים, והרדיוס של כיכר המחברת 6 רחובות הוא 6 מטרים. דוגמה: לפניכם סרטוט של רשת רחובות. בתוך כל כיכר מצוין הרדיוס של אותה הכיכר.



לפניכם רשת הרחובות NET בעיר מסוימת:



הולך רגל נדרש ללכת מכיכר אחת לאחרת במסלול הקצר ביותר. המסלול הקצר ביותר הוא המסלול שבו סכום הרדיוסים של כל הכיכרות שבהן הוא עובר הוא הקטן ביותר. סכום הרדיוסים אינו כולל את הרדיוס של הכיכר שבה הוא מתחיל את מסלולו אך הוא כולל את הרדיוס של הכיכר שבה הוא מסיים את מסלולו.

דוגמה: בסרטוט שלעיל של רשת הרחובות NET מודגש המסלול הקצר ביותר מן הכיכר A לכיכר E. גודל הרדיוסים במסלול זה הוא  $3+2+3$  וסכומם הוא 8. סכום זה הוא הקטן ביותר מבין כל האפשרויות.

א. ברשת הרחובות NET, מהו המסלול הקצר ביותר מכיכר H לכיכר C? כתבו את שמות הכיכרות במסלול זה, לפי הסדר, משמאל לימין (אין צורך לבצע מעקב).

ב. (1) כתבו אלגוריתם המוצא עבור רשת רחובות כלשהי את המסלול הקצר ביותר מן הכיכר  $K_1$  לכיכר  $K_2$ .

הערה: יש לכתוב אלגוריתם יעיל שאינו עובר על כל המסלולים האפשריים.

(2) סרטטו את הגרף המייצג את רשת הרחובות NET הנתונה לעיל, באופן שיתאים לאלגוריתם שכתבתם.

**מודלים חישוביים**

אם למדתם מסלול זה, ענו על אחת מן השאלות 9–10 (25 נקודות).

9. נתונות השפות  $L_4 - L_1$  מעל הא"ב  $\{a,b\}$ :

$L_1 =$  שפת כל המילים שבהן מספר המופעים של האות  $a$  שווה למספר המופעים של האות  $b$ .

$L_2 =$  שפת כל המילים שבהן מספר המופעים של האות  $a$  גדול ממספר המופעים של האות  $b$ .

$L_3 =$  שפת כל המילים שיש בהן יותר משלוש אותיות.

$L_4 =$  שפת כל המילים שמתחילות באות  $a$  ומסתיימות באות  $b$  או שמתחילות באות  $b$  ומסתיימות באות  $a$ .

ענו על כל הסעיפים א–ז שלפניכם:

א. בנו אוטומט סופי דטרמיניסטי המקבל את השפה  $L_4$ .

ב. הוכיחו שהשפה  $L_3 \cap L_4$  רגולרית.

ג. כתבו את השפה המתקבלת מן הפעולה  $L_1 \cup L_2$ . האם השפה המתקבלת רגולרית? נמקו את תשובתכם.

ד. כתבו את השפה המתקבלת מן הפעולות  $L_1 \cup L_2 \cup \bar{L}_2$ .

ה. האם השפה  $L_1 \cap L_2$  רגולרית? נמקו את תשובתכם.

ו. בנו אוטומט סופי דטרמיניסטי שאינו מלא, המקבל את השפה  $L_2 \cap \bar{L}_3$ .

ז. כתבו את השפה המתקבלת מן הפעולה  $L_4 \cap R(L_4)$ .

10. הפעולה **Generate** מבוצעת על שני מספרים שאורכם זהה, והם מורכבים אך ורק מן הספרות 0 ו-1. הפעולה יוצרת מילה חדשה באותו האורך, באופן שלהלן:

אם בשני המספרים הספרה במקום ה- $n$  היא 0, האות במקום ה- $n$  היא F, אחרת האות במקום ה- $n$  היא T.  
דוגמה: פעולת **Generate** על המספרים  $x = 11001$  ו- $y = 10010$ , יוצרת את המילה TTFTT כמתואר לפניכם:

	0	1	2	3	4
x	1	1	0	0	1
y	1	0	0	1	0
	<b>Generate</b>				
המילה המתקבלת	T	T	F	T	T

כתבו מכונת טיורינג המקבלת בתחילת הסרט קלט של שני מספרים שאורכם זהה, המורכבים אך ורק מן הספרות 0 ו-1. המכונה מחזירה את המילה המתקבלת מפעולת **Generate** על שניהם.

הנחיות:

- שני המספרים בקלט מופרדים ב-#.
- אין צורך לשמור על הקלט (אפשר לשנות את הקלט לסימנים שונים).
- המילה החדשה תופיע במקום כלשהו בסרט בין שני סימני \$.
- אין צורך לבדוק את תקינות הקלט.

דוגמה:

הסרט לפני ההרצה:

	0	1	2	3	4	5		0	1	2	3	4	5			
⊢	1	0	1	0	1	0	#	1	0	1	1	0	0	Δ	Δ	...

הסרט לאחר ההרצה:

...	\$	T	F	T	T	T	F	\$	...
-----	----	---	---	---	---	---	---	----	-----

**שימו לב: המשך המבחן בעמוד הבא.**

### תכנות מונחה עצמים בשפת Java

אם למדתם מסלול זה ואתם כותבים בשפת Java, ענו על אחת מן השאלות 11–12 (25 נקודות).

11. במסעדת "הרמה" אפשר להזמין מקום עבור סועד אחד או יותר. בסוף הארוחה, כל אחד מן הסועדים משלם בנפרד על מה שאכל. כל סועד יכול לבחור אם לשלם במזומן או באמצעות אפליקציה או בכרטיס אשראי. למשלמים בכרטיס אשראי יש אפשרות לפצל את החשבון לכמה תשלומים שווים.

כדי לנהל מערכת תשלומים, מפתחים עבור המסעדה פרויקט ובו הממשק **IPayment (Interface)**, והמחלקות **Restaurant, Reservation, Credit, App, Cash**, כמפורט להלן:

#### ❖ **IPayment** – תשלום

בממשק קיימת הפעולה: `double getPrice()`

#### ❖ **Cash** – תשלום במזומן

תכונות המחלקה:

- `sumCash` – הסכום לתשלום במזומן, מטיפוס ממשי.
- `name` – שם הלקוח, מטיפוס מחרוזת.

#### ❖ **App** – תשלום באמצעות אפליקציה

תכונות המחלקה:

- `sumApp` – הסכום לתשלום באפליקציה, מטיפוס ממשי.
- `phoneNumber` – מספר טלפון, מטיפוס מחרוזת.

#### ❖ **Credit** – תשלום בכרטיס אשראי

תכונות המחלקה:

- `num` – מספר התשלומים, מטיפוס שלם.
- `part` – הסכום בכל תשלום, מטיפוס ממשי.
- `creditNumber` – מספר כרטיס האשראי, מטיפוס מחרוזת.

#### ❖ **Reservation** – הזמנה

תכונות המחלקה:

- `date` – תאריך ההזמנה, מטיפוס מחרוזת.
- `total` – החשבון סך הכול של כל הסועדים יחד באותה הזמנה, מטיפוס ממשי.
- `payments` – מערך השומר את התשלום ששילם כל אחד מן הסועדים באותה הזמנה, מטיפוס **IPayment** (המערך ללא ערכי null).

#### ❖ **Restaurant** – המסעדה

תכונות המחלקה:

- `array` – מערך בגודל 10,000, מטיפוס **Reservation**.
  - `current` – כמות ההזמנות השמורות, מטיפוס שלם.
- הניחו שההזמנות נשמרות ברצף במערך ואין ביניהן ערכי null.
- הערה: הניחו שבכל מחלקה הוגדרו פעולות `get` ו-`set` ופעולות בונות.

(שימו לב: המשך השאלה בעמוד הבא.)

א. (1) סרטוט תרשים הייררכייה המתאר את הקשרים של המחלקות והממשק של הפרויקט.

יש לסמן מימוש ממשק באמצעות החץ <----- והכלה באמצעות הסימן ◆.

(2) המסעדה עושה הנחות לסועדים בהתאם לצורת התשלום: סועד שמשלם במזומן והסכום לתשלום הוא מעל 200 שקלים זכאי ל-10% הנחה, סועד שמשלם באמצעות האפליקציה זכאי תמיד ל-5% הנחה, וסועד שמשלם בכרטיס אשראי משלם מחיר מלא.

פעולת הממשק getPrice מחזירה את סך התשלום שכל סועד משלם לאחר שקלול ההנחה (כאשר אין הנחה, הפעולה מחזירה מחיר מלא). ממשו את פעולת הממשק getPrice בכל אחת מן המחלקות הנדרשות.

ב. כתבו במחלקה Reservation פעולה ששמה cashTotal. הפעולה מחזירה את סכום הכסף המזומן שהתקבל בהזמנה (לאחר שקלול ההנחה).

ג. לפניכם פעולה במחלקה Reservation:

```
public void printDetails () {
    for (int i = 0; i < payments.length; i++) {
        System.out.println (payments[i].getDetails());
    }
}
```

הפעולה מדפיסה עבור כל סועד פרטים מסוימים על פי הקריטריונים שלהלן:

- אם שילם הסועד במזומן, הפעולה מדפיסה את השם שלו.
- אם שילם הסועד באמצעות האפליקציה, הפעולה מדפיסה את מספר הטלפון שלו.
- אם שילם הסועד בכרטיס אשראי, הפעולה מדפיסה את מספר כרטיס האשראי שלו.

הוסיפו את הפעולות הנחוצות לפרויקט כדי שהפעולה תבצע את הנדרש. ציינו עבור כל פעולה שהוספתם לאיזו מחלקה או לאיזה ממשק היא שייכת.

הערה: אין לשנות את הפעולה printDetails.

12. לפניכם המחלקות First ו־ Second :

```
public class First
{
    private static int count = 0;
    protected int x;
    protected int y;
    public First (int num) {
        this.x = num;
        this.y = num;
        count ++;
        System.out.println ("First 1");
    }
    public First (int num1 , int num2) {
        this.x = num1;
        this.y = num2;
        count ++;
        System.out.println ("First 2");
    }
    public static int getCount() {
        return count;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    public int sum() {
        return this.x + this.y;
    }
    public void add(First other) {
        this.x += other.x;
        this.y += other.y;
        System.out.println("x = "+ this.x +
            "y = "+ this.y);
    }
}
```

```
public class Second extends First
{
    private int z;
    public Second (int num) {
        super (num);
        this.z = num;
        System.out.println ("Second");
    }
    public int sum() {
        return super.sum() + this.z;
    }
    public void add (First other) {
        this.x += other.getX();
        this.y += other.getY();
        if (other instanceof Second)
            this.z += ((Second)other).z;
        System.out.println("x = "+ this.x +
            " y = "+ this.y+ " z = "+ this.z);
    }
}
```

(שימו לב: המשך השאלה בעמוד הבא.)



נתונה המחלקה Tester :

```
public class Tester
{
    public static void main(String[] args)
    {
        First f1 = new First (40);
        First f2 = new First (40, 50);
        First f3 = new Second (100);
        Second s1 = new Second (100);
        Second s2 = new Second (100);
        // ***
    }
}
```

- א. ציירו את העצמים שנוצרו בפעולה main , וכתבו את הפלט של הפעולה.
- ב. הציבו כל אחת מן הפקודות 1-10 שלהלן בפעולה main במקום המצוין לעיל ב- \*\* .  
 כתבו במחברת את מספר הפקודה וציינו אם הקוד תקין או לא תקין.  
 אם הקוד תקין – כתבו את הפלט, ואם הוא אינו תקין, הסבירו מדוע.  
 הערה: אין קשר בין הפקודות. כלומר, יש להתייחס לכל פקודה כאילו היא היחידה בפעולה main .

1. System.out.println ("Total = " + First.getCount());
2. System.out.println ("Total = " + Second.getCount());
3. System.out.println ("sum = " + s1.sum());
4. System.out.println ("sum = " + f3.sum());
5. s1=new First (100);
6. f1.add (s2);
7. s1.add (s2);
8. s2.add (f3);
9. ((First)s1).add (f1);
10. s1=new Second (100, 100);

### תכנות מונחה עצמים בשפת C#

אם למדתם מסלול זה ואתם כותבים בשפת C#, ענו על אחת מן השאלות 13–14 (25 נקודות).

13. במסעדת "הרמה" אפשר להזמין מקום עבור סועד אחד או יותר. בסוף הארוחה, כל אחד מן הסועדים משלם בנפרד על מה שאכל. כל סועד יכול לבחור אם לשלם במזומן או באמצעות אפליקציה או בכרטיס אשראי. למשלמים בכרטיס אשראי יש אפשרות לפצל את החשבון לכמה תשלומים שווים. כדי לנהל מערכת תשלומים, מפתחים עבור המסעדה פרויקט ובו הממשק **IPayment (Interface)**, והמחלקות **Restaurant, Reservation, Credit, App, Cash**, כמפורט להלן:

#### ❖ **IPayment** – תשלום

בממשק קיימת הפעולה: `double GetPrice()`

#### ❖ **Cash** – תשלום במזומן

תכונות המחלקה:

- `sumCash` – הסכום לתשלום במזומן, מטיפוס ממשי.
- `name` – שם הלקוח, מטיפוס מחרוזת.

#### ❖ **App** – תשלום באמצעות אפליקציה

תכונות המחלקה:

- `sumApp` – הסכום לתשלום באפליקציה, מטיפוס ממשי.
- `phoneNumber` – מספר טלפון, מטיפוס מחרוזת.

#### ❖ **Credit** – תשלום בכרטיס אשראי

תכונות המחלקה:

- `num` – מספר התשלומים, מטיפוס שלם.
- `part` – הסכום בכל תשלום, מטיפוס ממשי.
- `creditNumber` – מספר כרטיס האשראי, מטיפוס מחרוזת.

#### ❖ **Reservation** – הזמנה

תכונות המחלקה:

- `date` – תאריך ההזמנה, מטיפוס מחרוזת.
- `total` – החשבון סך הכול של כל הסועדים יחד באותה הזמנה, מטיפוס ממשי.
- `payments` – מערך השומר את התשלום ששילם כל אחד מן הסועדים באותה הזמנה, מטיפוס **IPayment** (המערך ללא ערכי null).

#### ❖ **Restaurant** – המסעדה

תכונות המחלקה:

- `array` – מערך בגודל 10,000, מטיפוס **Reservation**.
  - `current` – כמות ההזמנות השמורות, מטיפוס שלם.
- הניחו שההזמנות נשמרות ברצף במערך ואין ביניהן ערכי null.
- הערה: הניחו שבכל מחלקה הוגדרו פעולות `Get` ו-`Set` ופעולות בונות.

(שימו לב: המשך השאלה בעמוד הבא.)

א. (1) סרטוט תרשים הייררכייה המתאר את הקשרים של המחלקות והממשק של הפרויקט.

יש לסמן מימוש ממשק באמצעות החץ <----- והכלה באמצעות הסימן ◆.

(2) המסעדה עושה הנחות לסועדים בהתאם לצורת התשלום: סועד שמשלם במזומן והסכום לתשלום הוא מעל 200 שקלים זכאי ל-10% הנחה, סועד שמשלם באמצעות האפליקציה זכאי תמיד ל-5% הנחה, וסועד שמשלם בכרטיס אשראי משלם מחיר מלא.

פעולת הממשק GetPrice מחזירה את סך התשלום שכל סועד משלם לאחר שקלול ההנחה (כאשר אין הנחה, הפעולה מחזירה מחיר מלא). ממשו את פעולת הממשק GetPrice בכל אחת מן המחלקות הנדרשות.

ב. כתבו במחלקה **Reservation** פעולה ששמה CashTotal. הפעולה מחזירה את סכום הכסף מזומן שהתקבל בהזמנה (לאחר שקלול ההנחה).

ג. לפניכם פעולה במחלקה **Reservation**.

```
public void PrintDetails () {
    for (int i = 0; i < payments.Length; i++) {
        Console.WriteLine (payments[i].GetDetails());
    }
}
```

הפעולה מדפיסה עבור כל סועד פרטים מסוימים על פי הקריטריונים שלהלן:

- אם שילם הסועד במזומן, הפעולה מדפיסה את השם שלו.
- אם שילם הסועד באמצעות האפליקציה, הפעולה מדפיסה את מספר הטלפון שלו.
- אם שילם הסועד בכרטיס אשראי, הפעולה מדפיסה את מספר כרטיס האשראי שלו.

הוסיפו את הפעולות הנחוצות לפרויקט כדי שהפעולה תבצע את הנדרש. ציינו עבור כל פעולה שהוספתם לאיזו מחלקה או לאיזה ממשק היא שייכת.

הערה: אין לשנות את הפעולה PrintDetails.

14. לפניכם המחלקות First ו־ Second :

```
public class First
{
    private static int count = 0;
    protected int x;
    protected int y;
    public First (int num) {
        this.x = num;
        this.y = num;
        count ++;
        Console.WriteLine ("First 1");
    }
    public First (int num1 , int num2) {
        this.x = num1;
        this.y = num2;
        count ++;
        Console.WriteLine ("First 2");
    }
    public static int GetCount() {
        return count;
    }
    public int GetX() {
        return x;
    }
    public int GetY() {
        return y;
    }
    public virtual int Sum() {
        return this.x + this.y;
    }
    public virtual void Add (First other) {
        this.x += other.x;
        this.y += other.y;
        Console.WriteLine ("x = "+this.x +
            " y = " +this.y);
    }
}
```

```
public class Second : First
{
    private int z;
    public Second (int num) : base (num) {
        this.z = num;
        Console.WriteLine ("Second");
    }
    public override int Sum() {
        return base.Sum() + this.z;
    }
    public override void Add (First other) {
        this.x += other.GetX();
        this.y += other.GetY();
        if (other is Second)
            this.z += ((Second)other).z;
        Console.WriteLine("x = " + this.x +
            " y = " + this.y + " z = " + this.z);
    }
}
```

(שימו לב: המשך השאלה בעמוד הבא.)

נתונה המחלקה Tester :

```
public class Tester
{
    public static void Main(string[] args)
    {
        First f1 = new First (40);
        First f2 = new First (40, 50);
        First f3 = new Second (100);
        Second s1 = new Second (100);
        Second s2 = new Second (100);
        // ***
    }
}
```

ציירו את העצמים שנוצרו בפעולה Main, וכתבו את הפלט של הפעולה.

הציבו כל אחת מן הפקודות 1–10 שלהלן בפעולה Main במקום המצוין לעיל ב- \*\*.

כתבו במחברת את מספר הפקודה וציינו אם הקוד תקין או לא תקין.

אם הקוד תקין – כתבו את הפלט, ואם הוא אינו תקין, הסבירו מדוע.

הערה: אין קשר בין הפקודות. כלומר, יש להתייחס לכל פקודה כאילו היא היחידה בפעולה Main.

1. Console.WriteLine ("Total = " + First.GetCount());
2. Console.WriteLine ("Total = " + Second.GetCount());
3. Console.WriteLine ("sum = " + s1.Sum());
4. Console.WriteLine ("sum = " + f3.Sum());
5. s1 = new First (100);
6. f1.Add (s2);
7. s1.Add (s2);
8. s2.Add (f3);
9. ((First)s1).Add (f1);
10. s1 = new Second (100, 100);

### בהצלחה!