

סוג הבחינה: מבחן מתכונת
מבחן: מתכונת 4
מספר השאלון: 899271
שם התלמיד/ה: _____

מדעי המחשב

מבני נתונים ומסלול - 4-5 יחידות לימוד

הוראות

א. משך הבחינה: שלוש שעות.

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

פרק ראשון - 2×25 - 50 נקודות

פרק שני - 2×25 - 50 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 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.

השאלות

פרק ראשון

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

1. (25 נקודות)

- במשחק קלפים נשמרות בתור הנקודות שצבר שחקן בכל סיבוב, לפי סדר הסיבובים. כל הערכים בתור הם מספרים שלמים חיוביים.
- "סיבוב מכריע" הוא איבר בתור שערכו שווה לסכום הערכים של כל האיברים שלפניו בתור.
- הערה:** האיבר הראשון בתור אינו "סיבוב מכריע".
- א.** כתבו פעולה ששמה isDecisive בשפת Java או IsDecisive בשפת C#, המקבלת תור - q מטיפוס שלם, ומספר שלם - m הגדול מ-0 וקטן או שווה לגודל התור.
- הפעולה תחזיר true אם האיבר במקום ה- m בתור הוא "סיבוב מכריע", אחרת היא תחזיר false.
- הערות:**
- בסיום הפעולה חובה לשמור על מבנה התור כפי שהתקבל.
 - אין להשתמש בסעיף זה במערך או ברשימה מקושרת. פתרון הכולל שימוש בהם לא יזוכה בנקודות.
- לדוגמה:** עבור התור שלפניכם:

סוף התור	1	2	3	4	5	6	7	ראש התור
	3	3	6	5	17	2	36	

- עבור $m = 1$ הפעולה תחזיר false (האיבר הראשון בתור אינו "סיבוב מכריע")
- עבור $m = 3$ הפעולה תחזיר true ($3 + 3 = 6$)
- עבור $m = 4$ הפעולה תחזיר false ($3 + 3 + 6 = 12 \neq 5$)
- עבור $m = 5$ הפעולה תחזיר true ($3 + 3 + 6 + 5 = 17$)
- ב.** כתבו פעולה ששמה atLeast בשפת Java או AtLeast בשפת C# המקבלת תור מטיפוס שלם - q , ומספר שלם k הגדול מ-0.
- הפעולה תחזיר true אם יש בתור לפחות k "סיבובים מכריעים". אחרת הפעולה תחזיר false.
- אפשר להשתמש בפעולה שכתבתם בסעיף א.

הערות:

- בפעולה זו אין צורך לשמור על התור שהתקבל.
- אין להשתמש בסעיף זה במערך או ברשימה מקושרת. פתרון הכולל שימוש בהם לא יזוכה בנקודות.

דוגמאות: עבור התור שבדוגמה שלעיל:

עבור $k = 4$ הפעולה תחזיר true מכיוון שיש בתור 4 "סיבובים מכריעים" (במקומות 2, 3, 5, 7).

עבור $k = 2$ הפעולה תחזיר true.

עבור $k = 5$ הפעולה תחזיר false.

2. (25 נקודות)

נתונה המחלקה **Runner** - רץ במועדון ריצה, ולה שלוש תכונות:

· name - שם הרץ, מטיפוס מחרוזת

· times - מערך מטיפוס שלם בגודל 30, המכיל את הזמנים (בדקות) של ריצות ה-10 ק"מ שהרץ השלים בחודש האחרון. בחודש אחד רץ משלים עד 30 ריצות, והזמנים נשמרים ברצף מתחילת המערך.

· amount - מספר הריצות שהרץ השלים בפועל בחודש האחרון, מטיפוס שלם. כל רץ השלים ריצה אחת לפחות.

הניחו שלתכונות המחלקה יש פעולות get/Set ו-set/Set.

א. ממשו את הפעולה שלפניכם השייכת לממשק המחלקה Runner:

Java - public int countBelow (int limit)

C# - public int CountBelow (int limit)

הפעולה מקבלת מספר שלם limit, ומחזירה את מספר הריצות של הרץ שהזמן שלהן קטן ממש מ-limit.

ב. נתון מערך arr מטיפוס Runner, ובו כל הרצים במועדון, ונתון מספר שלם limit. ריצה שהזמן שלה קטן מ-limit נקראת "ריצה מהירה".

כתבו פעולה חיצונית ששמה bestRunners בשפת Java או BestRunners בשפת C#, המקבלת את המערך arr ואת limit.

הפעולה מחזירה שרשרת חוליות מטיפוס מחרוזת, ובה השמות של הרצים שמספר הריצות המהירות שלהם הוא הגדול ביותר במועדון. אם יש כמה רצים כאלה, כולם יופיעו בשרשרת לפי סדר הופעתם במערך arr.

אם לאף רץ במועדון אין ריצה מהירה, הפעולה מחזירה null.

לדוגמה: במועדון ארבעה רצים, לפי סדרם במערך:

times	name
48, 55, 47, 49, 52	Dana
51, 46, 50	Omer
53, 60, 47, 49, 58, 45	Yael
57, 61	Noam

עבור limit = 50 הפעולה תחזיר שרשרת ובה Dana ואחריה Yael (לכל אחת מהן 3 ריצות מהירות, ולשאר פחות).

עבור limit = 45 הפעולה תחזיר null.

3. (25 נקודות)

נוסח Java

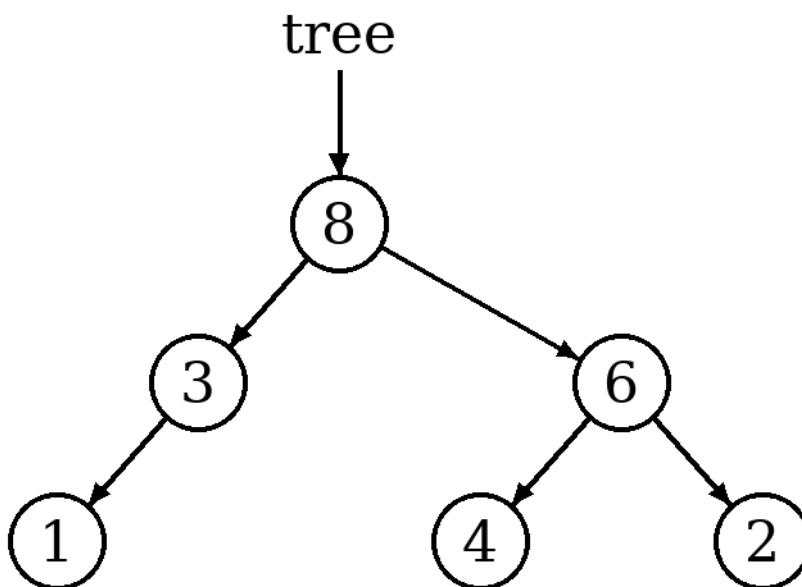
שימו לב: לשאלה זו שני נוסחים: בשפת Java ובשפת C#. ענו על הנוסח בשפה שבחרתם.

לפותרים בשפת Java

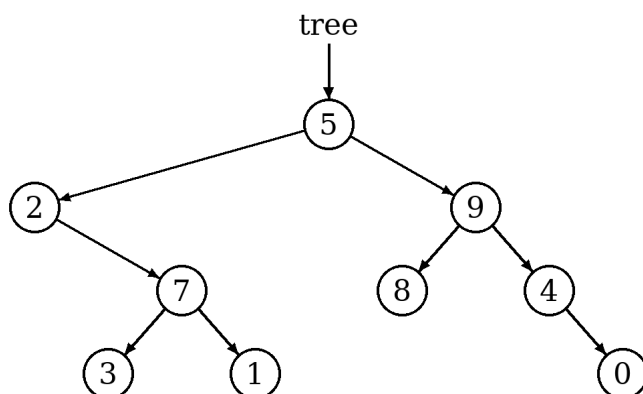
א. (1) נתונה הפעולה zig שלפניכם:

```
Java
public static int zig (BinNode<Integer> tree) {
    if (tree.getLeft() == null && tree.getRight() == null)
        return 1;
    int sum = 0;
    if (tree.getLeft() != null)
        sum = sum + zig (tree.getLeft());
    if (tree.getRight() != null)
        sum = sum - zig (tree.getRight());
    return sum;
}
```

בצעו מעקב אחרי הפעולה zig והעץ שלפניכם, וכתבו מה הפעולה מחזירה. יש להציג את המעקב.



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



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

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

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

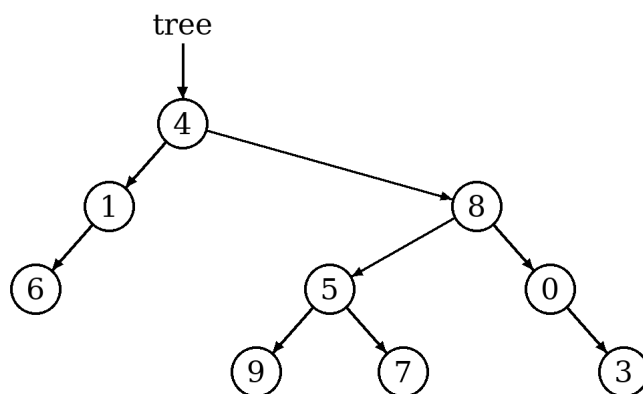
ב. (1) נתונה הפעולה zag שלפניכם:

Java

```
public static boolean zag (BinNode<Integer> tree) {
    if (tree == null)
        return false;
    if (zig (tree) < 0)
        return true;
    return zag (tree.getLeft()) || zag (tree.getRight());
}
```

בצעו מעקב אחרי הפעולה zag והעץ שלפניכם, וכתבו מה הפעולה מחזירה. יש להציג את המעקב.

בסעיף זה אין צורך להציג מעקב אחרי הפעולה zig.



(2) הציגו עץ שיש לו מעל 3 צמתים שבעבורו הפעולה zag תחזיר תוצאה הפוכה מן התוצאה שהתקבלה בתת-סעיף ב(1).

נוסח C#

שימו לב: לשאלה זו שני נוסחים: בשפת Java ובשפת C#. ענו על הנוסח בשפה שבחרתם.

לפותרים בשפת C#

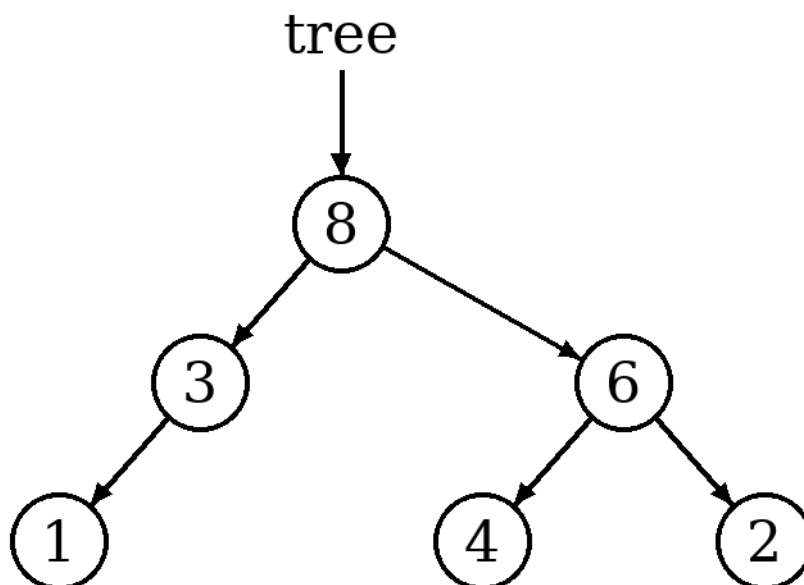
א. (1) נתונה הפעולה Zig שלפניכם:

C#

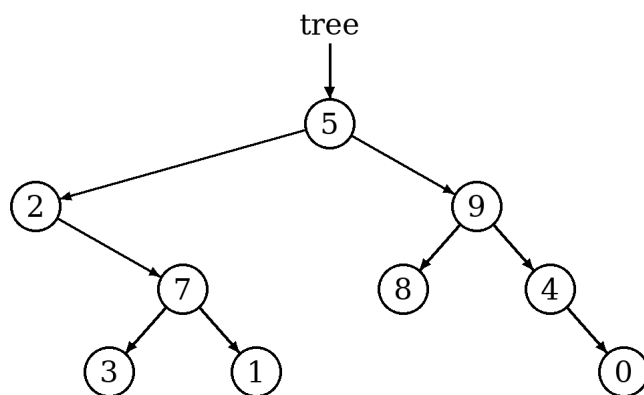
```
public static int Zig (BinNode<int> tree)
{
    if (tree.GetLeft() == null && tree.GetRight() == null)
        return 1;
    int sum = 0;
    if (tree.GetLeft() != null)
        sum = sum + Zig (tree.GetLeft());
    if (tree.GetRight() != null)
        sum = sum - Zig (tree.GetRight());
}
```

```
return sum;
}
```

בצעו מעקב אחרי הפעולה Zig והעץ שלפניכם, וכתבו מה הפעולה מחזירה. יש להציג את המעקב.



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



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

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

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

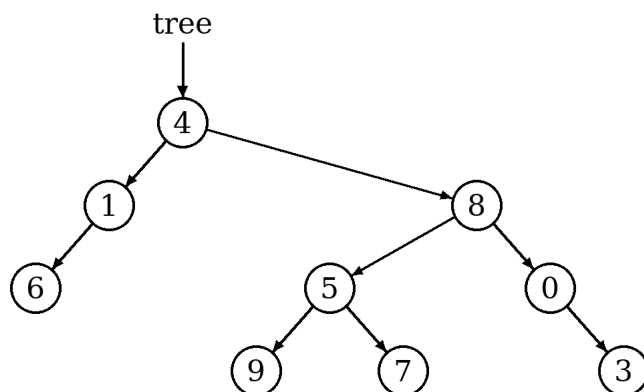
ב. (1) נתונה הפעולה Zag שלפניכם:

```
C#
public static bool Zag (BinNode<int> tree)
{
    if (tree == null)
        return false;
    if (Zig (tree) < 0)
        return true;
}
```

```
return Zag (tree.GetLeft()) || Zag (tree.GetRight());
}
```

בצעו מעקב אחרי הפעולה Zag והעץ שלפניכם, וכתבו מה הפעולה מחזירה. יש להציג את המעקב.

בסעיף זה אין צורך להציג מעקב אחרי הפעולה Zag.



(2) הציגו עץ שיש לו מעל 3 צמתים שבעבורו הפעולה Zag תחזיר תוצאה הפוכה מן התוצאה שהתקבלה בתת-סעיף ב(1).

פרק שני (50 נקודות) – אלגוריתמים

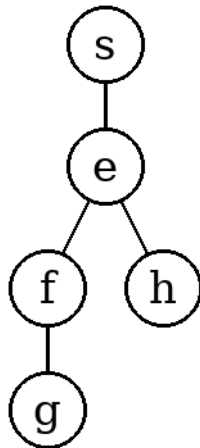
יש לענות על שתי שאלות במסלול שלמדתם (לכל שאלה – 25 נקודות).

4. (25 נקודות)

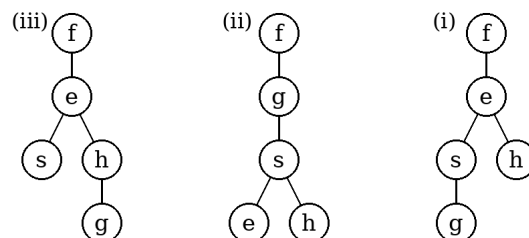
בשאלה זו שני סעיפים, א-ב, שאין קשר ביניהם. ענו על שני הסעיפים.

- א.** (1) לפניכם שש טענות 1-6. בחרו בארבע מהן, כתבו את מספר הטענה, וציינו בנוגע לכל טענה שבחרתם אם היא נכונה או לא נכונה. אם הטענה נכונה – נמקו מדוע, ואם היא אינה נכונה – הביאו דוגמה נגדית מגרף שיש בו 4 קודקודים לפחות.
- (1) נתון גרף לא מכוון ובו n קודקודים ו- m קשתות. בהכרח יש בגרף מעגל.
- (2) נתון גרף ממושקל, קשיר ולא מכוון, שבו כל משקלי הקשתות שונים זה מזה. הקשת השלישית הקלה ביותר בגרף (כלומר: יש בגרף בדיוק שתי קשתות קלות ממנה) נמצאת בהכרח בעץ הפורש המינימלי של הגרף.
- (3) נתון גרף לא מכוון וקשיר ובו n קודקודים. אם מוחקים מהגרף קשת אחת (ומשאירים את כל הקודקודים), יהיו בגרף לכל היותר 2 רכיבי קשירות.
- (4) נתון גרף לא מכוון. הרצנו BFS מקודקוד s וקיבלנו לכל קודקוד v את $d(v)$ – המרחק (מספר הקשתות במסלול הקצר ביותר) מ- s ל- v . לכל קשת $u-v$ בגרף שאחד מקצותיה ישיג מ- s , מתקיים $|d(u) - d(v)| \leq 1$.
- (5) נתון גרף מכוון. הרצנו BFS מקודקוד s וקיבלנו לכל קודקוד v את $d(v)$ – המרחק מ- s ל- v . לכל קשת $u \rightarrow v$ בגרף ששני קצותיה ישיגים מ- s , מתקיים $|d(u) - d(v)| \leq 1$.
- (6) נתון גרף לא מכוון. אם קיימת סריקת DFS מקודקוד s שבה התקבל עץ שהוא "שרשרת" (לכל קודקוד בעץ יש בן אחד לכל היותר), אז בכל סריקת DFS מקודקוד s יתקבל עץ שהוא שרשרת.
- ב.** נתון גרף לא מכוון G ובו 5 קודקודים: s, e, f, g, h .

לאחר הרצת DFS מקודקוד התחלה s התקבל העץ שבסרטוט הנתון:



בנוגע לכל אחד משלושת העצים שלפניכם, קבעו אם ייתכן שהוא התקבל מהרצת DFS על אותו הגרף G, מקודקוד התחלה f. נמקו את קביעותיכם.

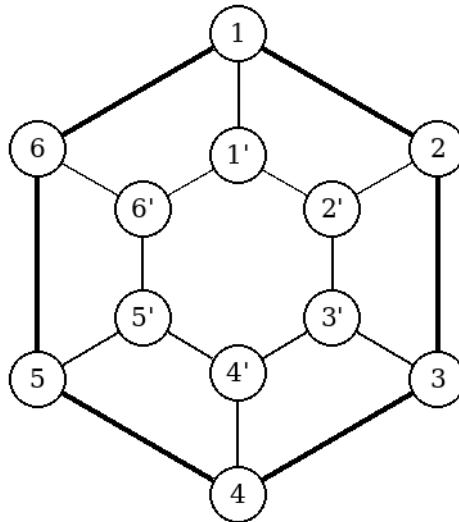


5. (25 נקודות)

- לפניכם שש טענות א-ו. בחרו **במש** מהן, וציינו בנוגע לכל טענה שבחרתם אם היא נכונה או לא נכונה. אם הטענה נכונה - נמקו מדוע, ואם הטענה לא נכונה - הביאו דוגמה נגדית.
- א.** בגרף G לא מכוון, אם סריקה לעומק (DFS) שמתחילה בקודקוד s מבקרת בכל הקודקודים של הגרף, אז גם סריקה לרוחב (BFS) שמתחילה בקודקוד s מבקרת בכל הקודקודים של הגרף.
- ב.** גרף לא מכוון שיש בו n קודקודים ו- n קשתות ($n \geq 3$) הוא בהכרח גרף קשיר.
- ג.** בגרף **מכוון** ממושקל שבו כל המשקלים חיוביים, המסלול הקל ביותר מקודקוד s אל קודקוד t הוא תמיד גם מסלול שמספר הקשתות בו הוא הקטן ביותר מבין כל המסלולים מ-s אל t.
- ד.** בגרף ממושקל, לא מכוון וקשיר, שבו יש **רק קשת אחת** שמשקלה הוא הקטן ביותר, הקשת הזאת שייכת לכל עץ פורש מינימלי של הגרף.
- ה.** האלגוריתם של דייקסטרה מחשב נכון את אורכי המסלולים הקלים ביותר מקודקוד המקור בכל גרף **מכוון** שאין בו מעגל שלילי, גם אם יש בו קשתות שמשקלן שלילי.
- ו.** נתון גרף מכוון G שיש בו לפחות שני קודקודים ויש לו מיון טופולוגי, ו- x ו- y הם הקודקוד הראשון והקודקוד האחרון במיון זה (בהתאמה). אם נוסיף ל-G את הקשת $x \rightarrow y$ (אם היא אינה קיימת בו), גם לגרף החדש יש מיון טופולוגי.

6. (25 נקודות)

גרף "מנסרה" - P_n הוא גרף לא מכוון שמורכב משני מעגלים של n קודקודים כל אחד: "המעגל החיצוני", שקודקודיו הם $1, 2, \dots, n$, ו"המעגל הפנימי", שקודקודיו הם $1', 2', \dots, n'$. בנוסף, כל קודקוד i של המעגל החיצוני מחובר בקשת לקודקוד i' של המעגל הפנימי (כך שסך הכול יש $2n$ קודקודים בגרף). מספר הקודקודים בכל מעגל גדול מ-2 (כלומר $n > 2$).
דוגמה: הגרף שלפניכם הוא גרף "מנסרה" - P_6 (קשתות המעגל החיצוני מודגשות).



"קשתות חיצוניות" הן הקשתות של המעגל החיצוני (למשל בדוגמה לעיל, הקשת שמחברת את הקודקודים 1 ו-2 זה לזה, הקשת שמחברת את הקודקודים 6 ו-1 זה לזה וכן הלאה).

"קשתות פנימיות" הן הקשתות של המעגל הפנימי (למשל בדוגמה לעיל, הקשת שמחברת את הקודקודים 1' ו-2' זה לזה).

"חישורים" הם הקשתות שמחברות קודקוד i לקודקוד i' (למשל בדוגמה לעיל, הקשת שמחברת את הקודקוד 3 לקודקוד 3').

א. (1) נתון גרף "מנסרה" - P_n .

(1) מה הן הדרגות של קודקודי המעגל החיצוני, ומה הן הדרגות של קודקודי המעגל הפנימי? נמקו את תשובתכם.

(2) כמה "קשתות חיצוניות", כמה "קשתות פנימיות" וכמה "חישורים" יש בגרף? נמקו את תשובתכם.

(3) בעבור אילו ערכים של n הגרף הוא דו-צדדי? נמקו את תשובתכם.

ב. נתון גרף ממושקל שהוא "מנסרה" - P_n , שבו לכל "הקשתות החיצוניות" ולכל "הקשתות הפנימיות" משקל 1, ולכל "החישורים" משקל 2. כמה עפ"מים (עצים פורשים מינימליים) שונים יש בגרף? נמקו את תשובתכם.

ג. (1) נתון גרף ממושקל שהוא "מנסרה" - P_n , שבו לכל "הקשתות החיצוניות" משקל a , לכל "הקשתות הפנימיות" משקל b , ולכל "החישורים" משקל s (כל המשקלים גדולים מ-0).

(1) הציבו ערכים ל- a, b ו- s , שבעבורם כל מסלול קצר (קל) בין שני קודקודים של המעגל החיצוני יעבור דרך קודקוד אחד לפחות של המעגל הפנימי. נמקו את תשובתכם.

(2) נתון כי $a = 10$ ו- $b = 2$. מהו הערך הגדול ביותר של s שבעבורו כל מסלול קצר (קל) בין שני קודקודים של המעגל החיצוני עדיין יעבור דרך קודקוד של המעגל הפנימי? נמקו את תשובתכם.

פרק שני (50 נקודות) - מודלים חישוביים

יש לענות על שתי שאלות במסלול שלמדתם (לכל שאלה - 25 נקודות).

7. (25 נקודות)

א. נתונה השפה L_1 מעל הא"ב $\{a, b, c\}$:

$$L_1 = \{b^n c^{2k} a^{n+k+2} \mid n, k \geq 1\}$$

א. (1) מהי המילה הקצרה ביותר בשפה L_1 ?

(2) בנו אוטומט מחסנית דטרמיניסטי המקבל את השפה L_1 .

ב. נגדיר את השפה L_2 מעל הא"ב $\{a, b, c\}$:

$$L_2 = L_1 \cdot L_1$$

בעבור כל אחת מן המילים שלפניכם, ציינו אם המילה שייכת לשפה L_2 , ונמקו.

$$(1) \text{ bccaaaabccaaaaa}$$

$$(2) \text{ bccaaabccaaaaa}$$

$$(3) \text{ bbccaaaaabccccaaaaa}$$

הערה: אין צורך לבנות אוטומט בעבור השפה L_2 .

8. (25 נקודות)

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

א. (1) נתונות שלוש שפות L_1, L_2, L_3 מעל הא"ב $\{a, b\}$. הסימון $R(L)$ מציין את השפה ההפוכה של L , הסימון L^* מציין את סגור קלייני של L (פעולת הכוכב), והסימון $L_1 - L_2$ מציין את ההפרש בין השפות (המילים של L_1 שאינן ב- L_2).

לפניכם שלוש טענות. בנוגע לכל אחת מן הטענות, קבעו אם היא נכונה או לא נכונה.

אם היא נכונה - הסבירו מדוע, ואם היא לא נכונה - הפריכו אותה באמצעות דוגמה נגדית: בחרו שפות ספציפיות והראו שעבורן הטענה לא מתקיימת.

$$(1) L_1^* \cup L_2 = (L_1 \cup L_2)^*$$

$$(2) R(L_1^*) = (R(L_1))^*$$

$$(3) L_3 = (L_1 \cdot L_3) - (L_2 \cdot L_3) \cdot (L_1 - L_2)$$

ב. נתונה השפה L מעל הא"ב $\{a, b\}$:

$$L = \{a^i b^j a^k \mid i \geq 1, k \geq 1, j = i + k\}$$

בנו אוטומט מחסנית דטרמיניסטי המקבל את השפה L .

9. (25 נקודות)

שזירה של שתי מילים x ו- y שאורכן זהה היא המילה שמתקבלת כשלוקחים לסירוגין אות מ- x ואות מ- y , משמאל לימין: האות הראשונה של x , האות הראשונה של y , האות השנייה של x , האות השנייה של y , וכן הלאה.

דוגמה: השזירה של $x = aab$ ושל $y = bba$ היא $ababba$:

2	1	0	
b	a	a	x
a	b	b	y

השזירה: $x_0 y_0 x_1 y_1 x_2 y_2 = a b a b b a$.

כתבו מכונת טיורינג המקבלת בתחילת הסרט שתי מילים שאורכן זהה מעל הא"ב $\{a, b\}$, ומחזירה את השזירה שלהן.

הנחיות:

- שתי המילים בקלט מופרדות ב- $\#$, ואורך כל אחת מהן 1 לפחות.
- אין צורך לשמור על הקלט (אפשר לשנות את הקלט לסימנים שונים).
- המילה המוחזרת תופיע במקום כלשהו בסרט בין שני סימני \$.
- אין צורך לבדוק את תקינות הקלט.

דוגמה:

הסרט לפני ההרצה:

...	Δ	Δ	a	b	b	#	b	a	a	⊖
-----	---	---	---	---	---	---	---	---	---	---

הסרט לאחר ההרצה:

...	\$	a	b	b	a	b	a	\$	...
-----	----	---	---	---	---	---	---	----	-----

כתבו מכונת טיורינג המקבלת בתחילת הסרט שתי מילים כמתואר לעיל, ומחזירה את השזירה שלהן בין שני סימני \$.

(25 נקודות)

תכנות מונחה עצמים

יש לענות על שתי שאלות במסלול שלמדתם (לכל שאלה - 25 נקודות).

10. (25 נקודות)

נוסח Java

לשאלה זו שני נוסחים: בשפת Java ובשפת C#.

לפותרים בשפת Java

בחברת השליחויות "מהר ובטוח" פותחה מערכת ממוחשבת ובה שומרים נתונים על המשלוחים. המערכת כוללת את המחלקות האלה: חבילה - **Parcel**, חבילה שבירה - **Fragile**, חבילה ובה כלי זכוכית - **Glass**, חבילה ובה מכשיר אלקטרוני - **Electronic**, מכתב - **Letter**. נתון חלק ממבנה המחלקות במערכת הממוחשבת:

Java

```
public class Parcel {
    private String sender;
    private int weight;
    public Parcel (String sender, int weight)
    {
        this.sender = sender;
        this.weight = weight;
    }
    public String toString() {
        return "From " + this.sender + ", " + weight + " kg.";
    }
}

public class Fragile {
    private String sender;
    private int weight;
    private boolean isInsured;    // האם החבילה מבוטחת
    public void warn () {
        System.out.println ("Careful! " + sender);
    }
}

public class Glass {
    private String sender;
    private int weight;
    private boolean isInsured;    // האם החבילה מבוטחת
    private int layers;          // מספר שכבות הריפוד
    public void warn () {
        System.out.println ("Careful! " + sender);
    }
}

public class Electronic {
    private String sender;
    private int weight;
    private boolean isInsured;    // האם החבילה מבוטחת
    private boolean hasBattery;   // האם יש במכשיר סוללה
    public void warn () {
        System.out.println ("Careful! " + sender);
    }
}

public class Letter {
    private String sender;
    private int weight;
    private boolean isRegistered; // האם זה דואר רשום
}
```

הניחו שיש פעולות get ו-set לתכונות המחלקות.

- א. כתבו מחדש את הכותרות, התכונות והפעולות של המחלקות: **Fragile**, **Glass**, **Electronic** ו-**Letter**, לפי עקרונות תכנות מונחה עצמים (אין לשנות את המחלקה **Parcel**). בסעיף זה אין צורך לכתוב פעולות בונות. הניחו שהפעולות **get** ו-**set** קיימות. נוסף על כך, סרטטו תרשים הייררכייה בין כל המחלקות. יש לסמן ירושה באמצעות החץ <---.
- ב. נתונה המחלקה **Tester**:

Java

```
public class Tester {
    public static void main (String[] args) {
        Parcel[] parcels = new Parcel [5];
        parcels [0] = new Glass ("Dana", 3, true, 4);
        parcels [1] = new Letter ("Omer", 1, true);
        parcels [2] = new Fragile ("Noa", 6, false);
        parcels [3] = new Electronic ("Yossi", 2, true, false);
        parcels [4] = new Letter ("Lior", 1, false);
        /**/
    }
}
```

הוסיפו פעולות בונות למחלקות **Fragile**, **Glass**, **Electronic** ו-**Letter**, המאתחלות את תכונות המחלקות (כך שהפעולה **main** תרוץ ללא שגיאות). בכל פעולה בונה סדר הפרמטרים הוא: קודם התכונות של **Parcel**, ואחריהן שאר התכונות לפי סדר הופעתן במחלקה הנתונה.

הערה: אין לשנות את הפעולה הבונה הנתונה של **Parcel**.

- ג. (1) בפעולה **main** נוסף קטע הקוד שלפניכם (במקום המסומן ב- *):

Java

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

קטע הקוד תקין ומדפיס את הפלט שלפניכם:

```
Glass from Dana! 4 layers
Registered letter. From Omer, 1 kg.
From Noa, 6 kg. Fragile!
From Yossi, 2 kg. Fragile!
Letter. From Lior, 1 kg.
```

הוסיפו את הפעולה **toString** במחלקות (רק היכן שיש צורך) לפי עקרונות תכנות מונחה עצמים, כך שהפלט יהיה מתאים.

הערה: אין לשנות את הפעולה **toString** במחלקה **Parcel**. תשובה שבה יהיה שינוי לא תזוכה בנקודות.

- (2) במחלקה **Tester** נוספה הפעולה שלפניכם:

Java

```
public static void alert (Parcel [] arr) {
    for (int i = 0; i < arr.length; i++) {
        arr[i].warn(); }
}
```

אם הפעולה תקינה, כתבו את הפלט שלה עבור המערך parcels (הנתון לעיל), ואם הפעולה לא תקינה - תקנו אותה (כך שהקריאה ל- warn בתוך הלולאה תהיה תקינה) וכתבו את הפלט עבור המערך parcels .

הערות:

- אין לשנות את המחלקות עצמן. תשובה שבה יהיה שינוי במחלקות לא תזוכה בנקודות.
- הפעולה צריכה לרוץ ללא שגיאות עבור כל מערך מטיפוס Parcel .

נוסח C#

לשאלה זו שני נוסחים: בשפת Java ובשפת #C.

לפותרים בשפת #C

בחברת השליחויות "מהר ובטוח" פותחה מערכת ממוחשבת ובה שומרים נתונים על המשלוחים. המערכת כוללת את המחלקות האלה: חבילה - **Parcel**, חבילה שבירה - **Fragile**, חבילה ובה כלי זכוכית - **Glass**, חבילה ובה מכשיר אלקטרוני - **Electronic**, מכתב - **Letter**. נתון חלק ממבנה המחלקות במערכת הממוחשבת:

```
C#
public class Parcel {
    private string sender;
    private int weight;
    public Parcel (string sender, int weight)
    {
        this.sender = sender;
        this.weight = weight;
    }
    public override string ToString() {
        return "From " + this.sender + ", " + weight + " kg.";
    }
}
public class Fragile {
    private string sender;
    private int weight;
    private bool isInsured;           // האם החבילה מבוטחת
    public void Warn () {
        Console.WriteLine ("Careful! " + sender);
    }
}
public class Glass {
    private string sender;
    private int weight;
    private bool isInsured;           // האם החבילה מבוטחת
    private int layers;               // מספר שכבות הריפוד
    public void Warn () {
        Console.WriteLine ("Careful! " + sender);
    }
}
public class Electronic {
    private string sender;
    private int weight;
    private bool isInsured;           // האם החבילה מבוטחת
    private bool hasBattery;          // האם יש במכשיר סוללה
    public void Warn () {
        Console.WriteLine ("Careful! " + sender);
    }
}
public class Letter {
    private string sender;
    private int weight;
    private bool isRegistered;        // האם זה דואר רשום
}
```

הניחו שיש פעולות Get ו-Set לתכונות המחלקות.

- א. כתבו מחדש את הכותרות, התכונות והפעולות של המחלקות: **Fragile**, **Glass**, **Electronic** ו- **Letter**, לפי עקרונות תכנות מונחה עצמים (אין לשנות את המחלקה **Parcel**). בסעיף זה אין צורך לכתוב פעולות בונות. הניחו שהפעולות **Get** ו- **Set** קיימות.
- נוסף על כך, סרטטו תרשים הייררכייה בין כל המחלקות. יש לסמן ירושה באמצעות החץ <---.
- ב. נתונה המחלקה **Tester**:

```
C#
public class Tester {
    public static void Main (string[] args) {
        Parcel[] parcels = new Parcel [5];
        parcels [0] = new Glass ("Dana", 3, true, 4);
        parcels [1] = new Letter ("Omer", 1, true);
        parcels [2] = new Fragile ("Noa", 6, false);
        parcels [3] = new Electronic ("Yossi", 2, true, false);
        parcels [4] = new Letter ("Lior", 1, false);
        /**
    }
}
```

הוסיפו פעולות בונות למחלקות **Fragile**, **Glass**, **Electronic** ו- **Letter**, המאתחלות את תכונות המחלקות (כך שהפעולה **Main** תרוץ ללא שגיאות). בכל פעולה בונה סדר הפרמטרים הוא: קודם התכונות של **Parcel**, ואחריהן שאר התכונות לפי סדר הופעתן במחלקה הנתונה.

- הערה: אין לשנות את הפעולה הבונה הנתונה של **Parcel**.
- ג. (1) בפעולה **Main** נוסף קטע הקוד שלפניכם (במקום המסומן ב- *):

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

קטע הקוד תקין ומדפיס את הפלט שלפניכם:

```
Glass from Dana! 4 layers
Registered letter. From Omer, 1 kg.
From Noa, 6 kg. Fragile!
From Yossi, 2 kg. Fragile!
Letter. From Lior, 1 kg.
```

הוסיפו את הפעולה **ToString** במחלקות (רק היכן שיש צורך) לפי עקרונות תכנות מונחה עצמים, כך שהפלט יהיה מתאים.

הערה: אין לשנות את הפעולה **ToString** במחלקה **Parcel**. תשובה שבה יהיה שינוי לא תזוכה בנקודות.

- (2) במחלקה **Tester** נוספה הפעולה שלפניכם:

```
C#
public static void Alert (Parcel [] arr) {
    for (int i = 0; i < arr.Length; i++) {
        arr[i].Warn(); }
}
```

אם הפעולה תקינה, כתבו את הפלט שלה עבור המערך parcels (הנתון לעיל), ואם הפעולה לא תקינה - תקנו אותה (כך שהקריאה ל-Warn בתוך הלולאה תהיה תקינה) וכתבו את הפלט עבור המערך parcels.

הערות:

- אין לשנות את המחלקות עצמן. תשובה שבה יהיה שינוי במחלקות לא תזוכה בנקודות.
- הפעולה צריכה לרוץ ללא שגיאות עבור כל מערך מטיפוס Parcel.

11. (25 נקודות)

נוסח Java

מרכז הספורט "קפיצה" בנה את המחלקה **Time** – שעה, שלה שתי תכונות מטיפוס שלם: שעה – **hour**, דקה – **minute**.

נוסף על כך, לצורך ניהול הפעילויות במרכז נבנתה המחלקה **Activity** – פעילות.

במרכז יש שלושה סוגי פעילויות: שיעורים קבוצתיים, השכרת מגרשים וטורנירים. כדי לשמור את המידע שבכל סוג פעילות, נבנו המחלקות האלה: שיעור קבוצתי – **Lesson**, השכרת מגרש – **Rental**, טורניר – **Tournament**.

התכונות של המחלקות הן בהתאם לדרישות המידע שלפניכם:

המידע ששומרים על שיעור קבוצתי – **Lesson**:

- **day** – היום בשבוע שבו מתקיים השיעור (1-7), מטיפוס שלם
- **start** – שעת ההתחלה, מטיפוס **Time**
- **coach** – שם המאמן, מטיפוס מחרוזת
- **maxParticipants** – מספר המשתתפים המרבי, מטיפוס שלם

המידע ששומרים על השכרת מגרש – **Rental**:

- **day** – היום בשבוע (1-7), מטיפוס שלם
- **start** – שעת ההתחלה, מטיפוס **Time**
- **renter** – שם השוכר, מטיפוס מחרוזת
- **court** – מספר המגרש המושכר, מטיפוס שלם

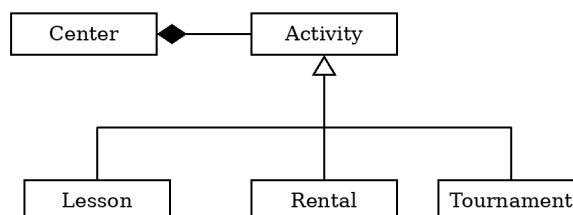
המידע ששומרים על טורניר – **Tournament**:

- **day** – היום בשבוע (1-7), מטיפוס שלם
- **start** – שעת ההתחלה, מטיפוס **Time**
- **teams** – מספר הקבוצות המשתתפות, מטיפוס שלם
- **courts** – מספרי המגרשים שבהם מתקיים הטורניר, במערך מטיפוס שלם (גודל המערך הוא לפי מספר המגרשים)

המידע ששומרים על פעילות – **Activity**:

אותן דרישות המידע המפורטות בסוגי הפעילויות שלעיל ושצריכות להישמר במחלקה **Activity** בהתאם לעקרונות תכנות מונחה עצמים.

נוסף על המחלקות האמורות, נבנתה המחלקה **Center** – מרכז הספורט, ולה שתי תכונות: **arr** – מערך בגודל 500 מטיפוס **Activity**, ו- **count** – מספר הפעילויות השמורות במערך. הפעילויות שמורות ברצף מתחילת המערך (בתאים 0 עד **count-1**), ואין ביניהן ערכי **null**. לפניכם תרשים חלקי של המחלקות:



שימו לב: ירושה מסומנת באמצעות החץ <--- (המשולש הריק בצד מחלקת-העל), והכלה מסומנת באמצעות הסימן <---◆ (המעוין המלא בצד המחלקה המכילה).

על הסעיפים שלפניכם יש לענות לפי עקרונות של תכנות מונחה עצמים.

א. העתיקו את התרשים למחברת והוסיפו את המחלקה Time לתרשים. כתבו את כותרת המחלקה פעילות – Activity ואת התכונות שלה.

הניחו שבשאר המחלקות כתובות התכונות המתאימות, ושכל המחלקות יש פעולות בונות ופעולות get ו-set.

ב. כתבו במחלקה Center פעולה ששמה lessonsBefore המקבלת מספר יום – day ועצם t – מטיפוס Time, ומחזירה מערך מטיפוס Lesson המכיל את כל השיעורים הקבוצתיים המתקיימים ביום day ומתחילים לפני השעה t. גודל המערך המוחזר צריך להיות שווה למספר השיעורים האלה (אם אין שיעורים כאלה, יוחזר מערך בגודל 0).

שימו לב:

הניחו שקיימת הפעולה שלפניכם במחלקה Time :

Java

```
public boolean before (Time other)
```

הפעולה מחזירה true אם השעה הנוכחית מוקדמת מהשעה other, אחרת (גם אם השעות שוות) היא מחזירה false. אפשר להשתמש בפעולה בלי לממש אותה.

ג. לפניכם כותרת הפעולה הפנימית – usesCourt במחלקה Activity :

Java

```
public boolean usesCourt (int court)
```

זימון הפעולה יחזיר true אם הפעילות (Activity) היא השכרה (Rental) של המגרש שמספרו court, או אם הפעילות היא טורניר (Tournament) שאחד המגרשים שלו הוא המגרש שמספרו court, אחרת יוחזר false (שיעורים קבוצתיים מתקיימים באולם ולא במגרש). ממשו את הפעולה, והוסיפו פעולות למחלקות המתאימות כדי לבצע את הנדרש. ציינו בעבור כל פעולה באיזו מחלקה היא מוספת.

הערה: בסעיף זה אין להשתמש בפעולה instanceof ובפעולות של המחלקה Object.

נוסח #C

מרכז הספורט "קפיצה" בנה את המחלקה **Time** – שעה, שלה שתי תכונות מטיפוס שלם: שעה – **hour**, דקה – **minute**.

נוסף על כך, לצורך ניהול הפעילויות במרכז נבנתה המחלקה **Activity** – פעילות.

במרכז יש שלושה סוגי פעילויות: שיעורים קבוצתיים, השכרת מגרשים וטורנירים. כדי לשמור את המידע שבכל סוג פעילות, נבנו המחלקות האלה: שיעור קבוצתי – **Lesson**, השכרת מגרש – **Rental**, טורניר – **Tournament**.

התכונות של המחלקות הן בהתאם לדרישות המידע שלפניכם:

המידע ששומרים על שיעור קבוצתי – **Lesson**:

- **day** – היום בשבוע שבו מתקיים השיעור (1-7), מטיפוס שלם
- **start** – שעת ההתחלה, מטיפוס **Time**
- **coach** – שם המאמן, מטיפוס מחרוזת
- **maxParticipants** – מספר המשתתפים המרבי, מטיפוס שלם

המידע ששומרים על השכרת מגרש – **Rental**:

- **day** – היום בשבוע (1-7), מטיפוס שלם
- **start** – שעת ההתחלה, מטיפוס **Time**
- **renter** – שם השוכר, מטיפוס מחרוזת
- **court** – מספר המגרש המושכר, מטיפוס שלם

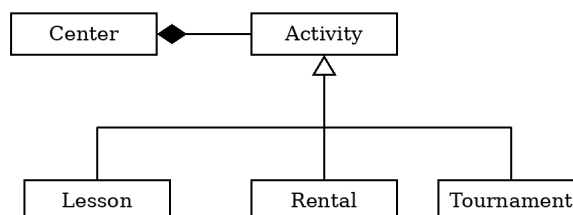
המידע ששומרים על טורניר – **Tournament**:

- **day** – היום בשבוע (1-7), מטיפוס שלם
- **start** – שעת ההתחלה, מטיפוס **Time**
- **teams** – מספר הקבוצות המשתתפות, מטיפוס שלם
- **courts** – מספרי המגרשים שבהם מתקיים הטורניר, במערך מטיפוס שלם (גודל המערך הוא לפי מספר המגרשים)

המידע ששומרים על פעילות – **Activity**:

אותן דרישות המידע המפורטות בסוגי הפעילויות שלעיל ושצריכות להישמר במחלקה **Activity** בהתאם לעקרונות תכנות מונחה עצמים.

נוסף על המחלקות האמורות, נבנתה המחלקה **Center** – מרכז הספורט, ולה שתי תכונות: **arr** – מערך בגודל 500 מטיפוס **Activity**, ו- **count** – מספר הפעילויות השמורות במערך. הפעילויות שמורות ברצף מתחילת המערך (בתאים 0 עד **count-1**), ואין ביניהן ערכי **null**. לפניכם תרשים חלקי של המחלקות:



שימו לב: ירושה מסומנת באמצעות החץ <--- (המשולש הריק בצד מחלקת-העל), והכלה מסומנת באמצעות הסימן <---◆ (המעוין המלא בצד המחלקה המכילה).

על הסעיפים שלפניכם יש לענות לפי עקרונות של תכנות מונחה עצמים.

א. העתיקו את התרשים למחברת והוסיפו את המחלקה Time לתרשים. כתבו את כותרת המחלקה פעילות – Activity ואת התכונות שלה.

הניחו שבשאר המחלקות כתובות התכונות המתאימות, ושכל המחלקות יש פעולות בונות ופעולות Get ו- Set.

ב. כתבו במחלקה Center פעולה ששמה LessonsBefore המקבלת מספר יום – day ועצם t – מטיפוס Time, ומחזירה מערך מטיפוס Lesson המכיל את כל השיעורים הקבוצתיים המתקיימים ביום day ומתחילים לפני השעה t. גודל המערך המוחזר צריך להיות שווה למספר השיעורים האלה (אם אין שיעורים כאלה, יוחזר מערך בגודל 0).

שימו לב:

הניחו שקיימת הפעולה שלפניכם במחלקה Time :

C#

```
public bool Before (Time other)
```

הפעולה מחזירה true אם השעה הנוכחית מוקדמת מהשעה other, אחרת (גם אם השעות שוות) היא מחזירה false. אפשר להשתמש בפעולה בלי לממש אותה.

ג. לפניכם כותרת הפעולה הפנימית – UsesCourt במחלקה Activity :

C#

```
public virtual bool UsesCourt (int court)
```

זימון הפעולה יחזיר true אם הפעילות (Activity) היא השכרה (Rental) של המגרש שמספרו court, או אם הפעילות היא טורניר (Tournament) שאחד המגרשים שלו הוא המגרש שמספרו court, אחרת יוחזר false (שיעורים קבוצתיים מתקיימים באולם ולא במגרש). ממשו את הפעולה, והוסיפו פעולות למחלקות המתאימות כדי לבצע את הנדרש. ציינו בעבור כל פעולה באיזו מחלקה היא מוספת.

הערה: בסעיף זה **אין** להשתמש בפעולות is ו- as ובפעולות של המחלקה object.

12. (25 נקודות)

נוסח Java

לפניכם המחלקות P, Q, R :

Java

```
public class P {
    public int n;

    public P() {
        this(5);
    }
    public P(int n) {
        this.n = n;
    }
    public int f(int k) {
        return k - this.n;
    }
    public int f(P other) {
        return this.n + other.n;
    }
}
```

Java

```
public class Q extends P {
    public Q() {
        this.n = this.n * 2;
    }
    public Q(int k) {
        super(k + 1);
    }
    public int f(P other) {
        return this.n * other.n;
    }
    public int f(Q other) {
        return other.n - this.n;
    }
}
```

Java

```
public class R extends Q {
    public R(int k) {
        super(k);
        this.n = this.n * 3 + f(2);
    }
    public int f(Q other) {
        return this.n - 2 * other.n;
    }
}
```

א. לפניכם קטע קוד שרץ ללא שגיאות:

Java

```
P p = new P();
P pq = new Q();
P pr = new R(4);
Q q = new Q(2);
Q qr = new R(0);
R r = new R(1);
```

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

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

ציינו את מספר שורת ההדפסה, וכתבו מה הקוד מדפיס.

Java

```
1. System.out.println(p.f(q));
2. System.out.println(pq.f(p));
3. System.out.println(pq.f(qr));
4. System.out.println(q.f(qr));
5. System.out.println(qr.f(q));
6. System.out.println(qr.f(pr));
7. System.out.println(r.f(r));
8. System.out.println(((Q)pr).f((Q)pq));
9. System.out.println(pr.f(pr.f(20)));
10. System.out.println(r.f(p));
```

נוסח #C

לפניכם המחלקות P, Q, R :

C#

```
public class P {
    public int n;

    public P() : this(5) { }
    public P(int n)
    {
        this.n = n;
    }
    public int F(int k)
    {
        return k - this.n;
    }
    public virtual int F(P other)
    {
        return this.n + other.n;
    }
}
```

C#

```
public class Q : P {
    public Q()
    {
        this.n = this.n * 2;
    }
    public Q(int k) : base(k + 1) { }
    public override int F(P other)
    {
        return this.n * other.n;
    }
    public virtual int F(Q other)
    {
        return other.n - this.n;
    }
}
```

C#

```
public class R : Q {
    public R(int k) : base(k)
    {
        this.n = this.n * 3 + F(2);
    }
    public override int F(Q other)
    {
        return this.n - 2 * other.n;
    }
}
```

א. לפניכם קטע קוד שרץ ללא שגיאות:

C#

```
P p = new P();
P pq = new Q();
P pr = new R(4);
Q q = new Q(2);
```

```
Q qr = new R(0);
R r = new R(1);
```

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

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

ציינו את מספר שורת ההדפסה, וכתבו מה הקוד מדפיס.

C#

```
1. Console.WriteLine(p.F(q));
2. Console.WriteLine(pq.F(p));
3. Console.WriteLine(pq.F(qr));
4. Console.WriteLine(q.F(qr));
5. Console.WriteLine(qr.F(q));
6. Console.WriteLine(qr.F(pr));
7. Console.WriteLine(r.F(r));
8. Console.WriteLine(((Q)pr).F((Q)pq));
9. Console.WriteLine(pr.F(pr.F(20)));
10. Console.WriteLine(r.F(p));
```

כל הבגרויות במדעי המחשב עם פתרונות מלאים: www.guyai.co.il/bagrut/computer-science