מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: **מועד חלופי**, קיץ תשפ״ו
מספר השאלון: 899271

מדעי המחשב

הוראות

א. <u>משך הבחינה</u>: שלוש שעות וחצי.

ב. <u>מבנה השאלון ומפתח ההערכה</u>: בשאלון זה שני פרקים.

יש לבחור בארבע שאלות סך הכול. לכל שאלה – 27 נקודות; תלמיד שצבר מעל 100 נקודות יקבל 100.

סך הכול – 100 נקודות.

בפרק השני יש לענות על שאלות במסלול אחד בלבד.

ג. <u>חומר עזר מותר בשימוש</u>: כל חומר עזר, חוץ ממחשבון שיש בו אפשרות תכנות.

ד. <u>הוראות מיוחדות</u>:

את כל התוכניות שיש לכתוב בשפת מחשב כתבו בשפה אחת בלבד – Java או C# .

<u>הערה</u>: לא יורדו נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.

יש לכתוב <u>במחברת הבחינה בלבד</u>, 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 4 שאלות מתוך 6 השאלות במבחן.

הערה: בכל שאלה בה נדרש קלט, אין צורך לבדוק את תקינות הקלט.

לפותרים בשפת Java: בכל שאלה בה נדרש קלט, הניחו שבתוכנית כתובה ההוראה:

```
Scanner scan = 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

1. נתונה המחלקה **DigitCount**, ולה שתי תכונות:

הסבר	התכונה
ערך סיפרה (0-9)	private int digit;
מספר מופעים של הספרה	private int count;

הניחו שיש פעולות get/Get ו – set/Set לתכונות המחלקה.

כמו כן קיימת במחלקה הפעולה הבאה:

הסבר	הפעולה

פעולה בונה	public DigitCount (int digit, int count) {
	this.digit = digit;
	this.count = count;
	}

א. ממשו את הפעולה שלפניכם:

Java: public static Node<DigitCount> buildList (int num)

C#: public static Node<DigitCount> BuildList (int num)

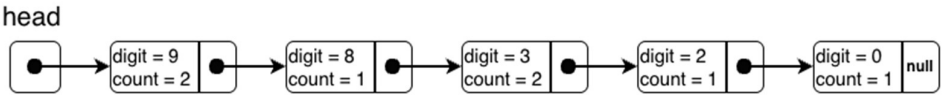
הפעולה מקבלת מספר שלם חיובי num הגדול מ-0 ומחזירה רשימה מקושרת מסוג DigitCount. עבור כל ספרה (0-9) במספר num יישמר ברשימה עצם מן המחלקה DigitCount שערכיו הספרה עצמה ומספר המופעים שלה במספר num. הרשימה תורכב לפי הכללים האלו:

- הרשימה תהיה ממוינת לפי ערך הספרות בסדר יורד (מהספרה בעלת הערך הגבוה לספרה בעלת הערך הנמוך).
- כל ספרה תופיע פעם אחת בלבד ברשימה.
- רק הספרות שמופיעות במספר num יופיעו ברשימה.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

דוגמה: עבור זימון הפעולה עם num = 3389920 וגם עבור num = 9389302 תוחזר הרשימה:



הסבר: ברשימה יש חוליה עבור כל ספרה שמופיעה במספר num. החוליה מייצגת את הספרה ואת מספר המופעים של הספרה ב-num. הרשימה כוללת רק ספרות שקיימות ב-num, והיא ממוינת לפי ערך הספרות מהגדול לקטן.

ב. ממשו את הפעולה שלפניכם:

Java: public static int buildMaxNum (int num)
C#: public static int BuildMaxNum (int num)

הפעולה מקבלת מספר שלם num הגדול מ-0 ומחזירה את המספר הגדול ביותר שניתן ליצור באמצעות הספרות במספר num, לפי מספר המופעים של כל ספרה.
הערה: ניתן להשתמש בפעולה שכתבתם בסעיף א.

דוגמה: עבור num = 3389920 וגם עבור num = 9389302 הפעולה תחזיר את המספר: 9983320.
הסבר: מספר זה הוא המספר המקסימלי שניתן ליצור באמצעות שינוי סדר הספרות במספר num.

/המשך בעמוד הבא/

2. נתונה המחלקה **Cube** - קוביה, ולה שתי תכונות:
   - size –גודל הקוביה בסמ״ר, מטיפוס מספר שלם.
   - type – סוג הקוביה, מטיפוס תו.

   הניחו שיש פעולות get/Get לתכונות המחלקה.

נתונה המחלקה **Box** - קופסה, ולה שתי תכונות:
- cubes - מערך בגודל 3 מטיפוס **Cube**, בו שומרים את הקוביות.
  ניתן להכניס למערך קוביות או שכולן מאותו סוג או שכולן מאותו גודל.
- count – כמות הקוביות במערך.

לפניכם פעולות המחלקה Box. ניתן להשתמש בפעולות ללא צורך במימוש:

הסבר	הפעולה

פעולה בונה.	public Box(){
	cubes= new Cube[3];
	count = 0;
	}

הפעולה מקבלת עצם מסוג Cube. אם ניתן להכניס את העצם למערך (בהתאם לדרישות לעיל), הפעולה מכניסה אותה למקום הראשון שפנוי במערך (ומעדכנת את ערך המשתנה count) ומחזירה true. אחרת (אם המערך מלא או שהעצם אינו עונה על הדרישות לעיל), הפעולה מחזירה false ללא שינוי ערכי המערך.	**Java:** public boolean add(Cube c) **C#:** public bool Add(Cube c)

א. קופסה נקראת ״מושלמת״ אם יש בה 3 קוביות מאותו הסוג וגם מאותו הגודל.
ממשו את הפעולה הפנימית שלפניכם במחלקה Box:

Java: public boolean isPerfect ()

C#: public bool IsPerfect ()

הפעולה מחזירה true אם הקופסה מושלמת. אחרת, מחזירה false.

ב. במפעל צעצועים אורזים קוביות לקופסאות לפי סדר ההגעה שלהן.

(1) ממשו את הפעולה החיצונית שלפניכם:

Java: public static Box packBox (Queue<Cube> q)

C#: public static Box PackBox (Queue<Cube> q)

הפעולה מקבלת תור של קוביות ומחזירה עצם מסוג Box. הפעולה תכניס קוביות מהתור (לפי סדר הופעתן) לקופסה, כל עוד פעולת ההכנסה אפשרית. אם לא ניתן להכניס את הקובייה הבאה בתור, הפעולה תעצור ותחזיר את הקופסה שנוצרה.

הערה: בסיום הפעולה סדר הקוביות בתור המקורי נשמר, ללא הקוביות שהוכנסו לקופסה.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

<u>דוגמה</u>: עבור התור שלפניכם:

ראש התור סוף התור

Cube	Cube	Cube	Cube	Cube	Cube	Cube	Cube
size= 1 type= ‘a’	size= 2 type= ‘a’	size= 2 type= ‘b’	size= 1 type= ‘a’	size= 1 type= ‘a’	size= 1 type= ‘a’	size= 3 type= ‘b’	size=3 type= ‘a’

הפעולה תחזיר עצם מסוג Box ובו הערך של count הוא 2 והמערך cubes הוא:

Cube	Cube	null
size= 1 type= ‘a’	size= 2 type= ‘a’	

מצב התור בסוף הפעולה הוא:

ראש התור סוף התור

Cube	Cube	Cube	Cube	Cube	Cube
size= 2 type= ‘b’	size= 1 type= ‘a’	size= 1 type= ‘a’	size= 1 type= ‘a’	size= 3 type= ‘b’	size=3 type= ‘a’

<u>הסבר</u>: הסוג של שתי הקוביות הראשונות זהה ולכן ניתן להכניסן לקופסה. הקוביה השלישית מסוג שונה ולכן לא נכנסת לקופסה.

(2) כתבו פעולה חיצונית בשם packAll בשפת Java או PackAll בשפת #C המקבלת תור q של עצמים מסוג Cube.

הפעולה:

- תחזיר תור מסוג Box לאחר הכנסת כל הקוביות לקופסאות לפי הכללים הנתונים לעיל.
- תדפיס perferct לאחר הכנסת כל הקוביות לקופסאות, אם קיימת לפחות קופסה אחת ״מושלמת״, אחרת, תדפיס not perfect.

<u>דוגמה</u>: עבור התור מסעיף א הפעולה תחזיר את התור שלפניכם ותדפיס perfect.

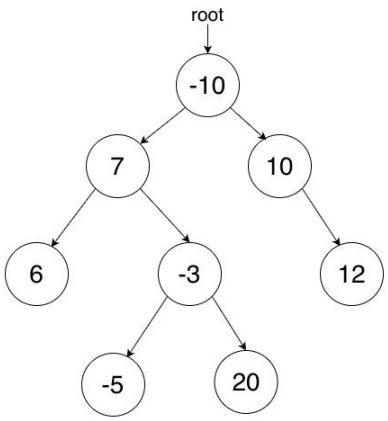
ראש התור סוף התור

Box			Box			Box			Box		
cubes			cubes			cubes			cubes		
size= 1 type= ‘a’	size= 2 type= ‘a’	null	size= 2 type= ‘b’	null	null	size= 1 type= ‘a’	size= 1 type= ‘a’	size= 1 type= ‘a’	size= 3 type= ‘b’	size=3 type= ‘a’	null
count = 2			count = 1			count = 3			count = 2		

<u>הסבר</u>: בהתאם לדרישות האמורות לעיל, יש צורך בארבע קופסאות. הקופסה השלישית בתור היא קופסה מושלמת.

/המשך בעמוד הבא/

3. לפניכם עץ בינארי ששורשו root המכיל מספרים שלמים:



א. נתונה הפעולה something בשפת Java ו- Something בשפת C#:

Java	public static int something (BinNode<Integer> root) {
	if (root == null)
	return 0;
	if (root.getLeft() == null && root.getRight() == null)
	return root.getValue ();
	int ans1 = something(root.getLeft());
	int ans2 = something(root.getRight());
	if (ans1 > ans2)
	return ans1 + root.getValue();
	return ans2 + root. getValue();
	}

C#	public static int Something (BinNode<int> root) {
	if (root == null)
	return 0;
	if (root.GetLeft() == null && root.GetRight() == null)
	return root.GetValue();
	int ans1 = Something(root.GetLeft());
	int ans2 = Something(root.GetRight());
	if (ans1 > ans2)
	return ans1 + root.GetValue();
	return ans2 + root. GetValue();
	}

(1) מהו הערך שתחזיר הפעולה עבור הזימון: something (root) בשפת Java או Something (root) בשפת C# עם העץ הנתון? יש להציג מעקב על ביצוע הפעולה.

(2) הסבירו מה מבצעת הפעולה עבור כל עץ של מספרים שלמים.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

ב. נתונה הפעולה sod בשפת Java ו- Sod בשפת C# המקבלת עץ בינארי של מספרים שלמים, בתחום הערכים 100- עד 100:

Java	public static int sod (BinNode<Integer> root) {
	if (root == null)
	return -101;
	int x = something (root);
	int left = sod (root.getLeft());
	int right = sod (root.getRight());
	return Math.max (x, Math.max (left, right));
	}

C#	public static int Sod (BinNode<int> root) {
	if (root == null)
	return -101;
	int x = Something (root);
	int left = Sod (root.GetLeft());
	int right = Sod (root.GetRight());
	return Math.Max (x, Math.Max (left, right));
	}

מהו הערך שתחזיר הפעולה עבור הזימון: sod (root) בשפת Java או Sod (root) בשפת C# עם העץ הנתון לעיל? יש להציג מעקב על ביצוע הפעולה.
אין צורך לבצע מעקב אחרי הפעולה something/Something אלא רק לציין את הערך שחוזר מהפעולה.

/המשך בעמוד הבא/

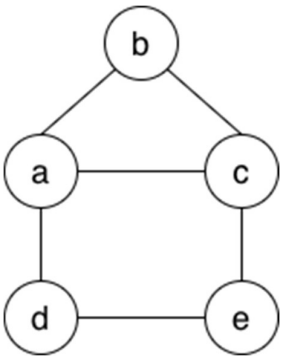
פרק שני

בפרק זה שאלות בשלושה מסלולים:
אלגוריתמים, שאלות 4-6
מודלים חישוביים, שאלות 7-9
תכנות מונחה עצמים בשפת Java ובשפת C# , שאלות 10-12.
ענו על שאלות <u>מתוך מסלול אחד בלבד</u>.

<u>אלגוריתמים</u>

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

א. לפניכם גרף לא מכוון $G=(V, E)$.



(1) האם הגרף קשיר? נמקו.
(2) האם הגרף מכיל מעגל? אם כן, ציינו מעגל אחד בגרף. אם לא, נמקו.
(3) אם הגרף דו־צדדי, הציגו חלוקה אפשרית (ציינו את הצמתים בכל צד).
אם הגרף אינו דו־צדדי, הסירו <u>מספר מינימלי</u> של קשתות כך שהגרף שיתקבל יהיה דו־צדדי, והציגו חלוקה אפשרית לשתי קבוצות.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

ב. נתון גרף לא מכוון $G=(V,E)$ המיוצג ע״י מטריצת הסמיכויות הבאה:

	a	**b**	**c**	**d**	**e**	**f**
a	0	1	1	0	0	0
b	1	0	0	1	0	0
c	1	0	0	1	1	0
d	0	1	1	0	0	0
e	0	0	1	0	0	1
f	0	0	0	0	1	0

(מקרא: 1 – קיימת קשת, 0 – לא קיימת קשת.)

(1) שרטטו את הגרף G המיוצג על ידי המטריצה הנתונה.

(2) כמה רכיבי קשירות יש בגרף G? נמקו.

(3) הציגו את המסלול הקצר ביותר מהקודקוד a לקודקוד f.

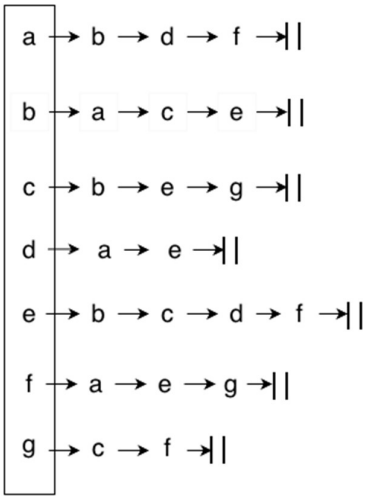
(4) האם קיימת קשת שהסרתה מהגרף G, תגדיל את מספר רכיבי הקשירות ב-1? אם כן, ציינו את הקשת.

(5) האם קיימת קשת שהסרתה מהגרף G, לא תשנה את מספר רכיבי הקשירות בגרף? אם כן, ציינו את הקשת.

/המשך בעמוד הבא/

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

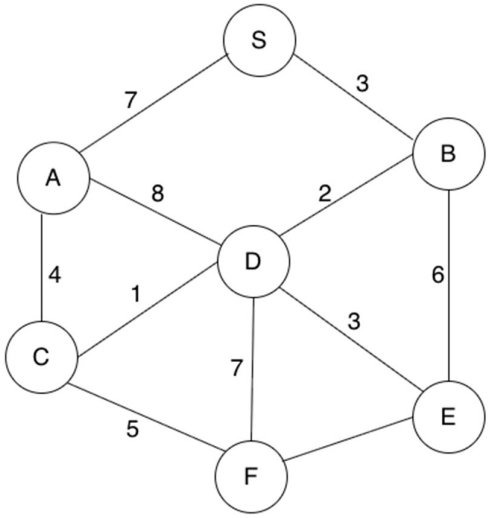
א. נתון גרף לא מכוון $G=(V,E)$, המיוצג על ידי רשימת הסמיכויות שלפניכם.



(1) הפעילו אלגוריתם סריקה לעומק DFS על הגרף הנתון, החל מהקודקוד a. שרטטו במחברתכם את עץ הסריקה DFS שמתקבל.

(2) הפעילו אלגוריתם סריקה לרוחב BFS על הגרף הנתון, החל מהקודקוד a. שרטטו במחברתכם את עץ הסריקה BFS שמתקבל.

ב. לפניכם גרף לא מכוון ממושקל G. לכל קשת יש משקל חיובי.



(1) כתבו את המסלולים הקצרים (הקלים) בגרף G, מקודקוד S לכל אחד מהקודקודים בגרף. אין צורך להציג מעקב.

(2) בעבור הגרף G הנתון, שרטטו את עץ המסלולים הקצרים מקודקוד S.

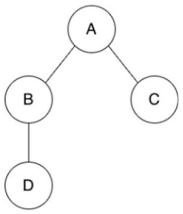
/המשך בעמוד הבא/

6. לפניכם שני סעיפים, א–ב, שאין קשר ביניהם. ענו על שני הסעיפים.

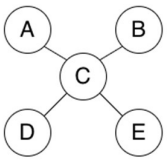
א. לפניכם שבע טענות 1-7 בחרו <u>בחמש</u> מהן וכתבו את מספריהם. ציינו בנוגע לכל טענה שבחרתם האם היא נכונה או לא נכונה. אם הטענה נכונה – נמקו מדוע, ואם היא אינה נכונה – הביאו דוגמה נגדית מגרף שיש בו <u>ארבעה קודקודים לפחות</u>.

(1) בגרף לא מכוון שבו לכל קודקוד דרגה זוגית הגדולה מ-0, קיים מעגל.
(2) בגרף לא מכוון קשיר, הסרת קשת שאינה שייכת לאף מעגל תגרום לגרף להיות לא קשיר.
(3) כל עץ הוא גרף דו צדדי וגם כל גרף דו צדדי הוא עץ.
(4) אם בגרף מכוון קיים מעגל, אז הגרף קשיר חזק (היטב).
(5) אם גרף לא מכוון הוא עץ, אז בין כל שני קודקודים בגרף קיים מסלול יחיד.
(6) בגרף לא מכוון ללא מעגלים, אם מוסיפים קשת אחת כלשהי, בהכרח ייווצר מעגל.
(7) נתון גרף לא מכוון ובו n קודקודים. אם יש בגרף n-1 קשתות, בהכרח הוא עץ.

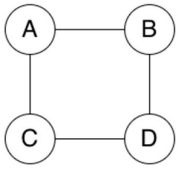
ב. עץ נקרא ״**עץ רדוד**״ אם מכל צומת בעץ יש מרחק של לכל היותר 3 קשתות לכל צומת אחר בעץ.
<u>דוגמה ל״עץ רדוד״</u>:



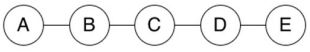
גרף לא מכוון וקשיר נקרא ״**גרף BFS רדוד**״ אם ב<u>כל</u> הרצת BFS על הגרף, מ<u>כל</u> קודקוד התחלה שנבחר, מתקבל ״עץ רדוד״.
לפניכם שישה גרפים G1 – G6. בחרו <u>בארבעה</u> מהם, וכתבו בנוגע לכל גרף שבחרתם אם הוא ״גרף BFS רדוד״ או לא. אם כתבתם שלא – הציגו סריקת BFS שבעבורה מתקבל עץ <u>שאינו</u> ״רדוד״.



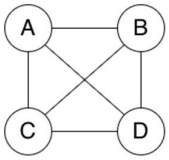
G1



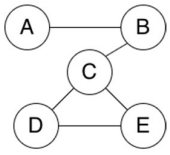
G2



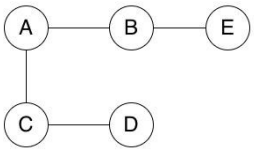
G3



G4



G5



G6

/המשך בעמוד הבא/

<u>מודלים חישוביים</u>

7. לפניכם שני סעיפים, א-ב שאינם קשורים זה לזה. עליכם לענות על שניהם.

א. לפניכם השפות הרגולריות L_1 ו- L_2 מעל האי״ב $\{a,b\}$:

$L_1 = \{a^{2k} \mid k>=0\}$

$L_2 = \{b^{2k+1} \mid k>=0\}$

הוכיחו בעזרת תכונות הסגירות של השפות הרגולריות שהשפה L_3 רגולרית.

$L_3 = \{a^n\, b^m \mid n,m>=0,\ m\%2 \neq n\%2\}$

ב. נתונות השפות הבאות:

$L = \{a^z w_1 w_2 \ldots w_k d^{k+1} \mid z\%2=1,\ w_i \in L_1,\ k>0\}$

$L_1 = \{b^n c^m \mid n,m>0\}$

<u>דוגמה למילה בשפה L</u>: aaabbcbccddd

a^z	w_1	w_k	d^{k+1}
aaa	bbc	bcc	ddd

<u>הסבר</u>: מספר תווי a הוא אי זוגי. בין תווי a בתחילת המילה לתווי d בסוף המילה, יש שתי מילים השייכות לשפה L_1, לכן k=2 ובהתאמה מספר תווי d בסוף המילה הוא 3.

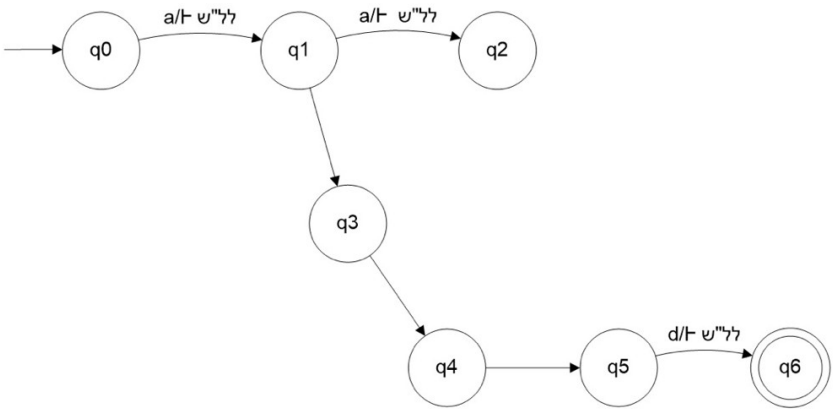
(1) כתבו את המילה הקצרה ביותר בשפה L.

(2) נתון אוטומט מחסנית דטרמינסטי חלקי המקבל את השפה L. האוטומט כולל את כל המצבים (כולל סימון מצב מקבל).

עליכם להשלים את המעברים החסרים ואת פירוט המעברים הקיימים (התו במעבר, הסימן בראש המחסנית והפעולה על המחסנית).

<u>הערה</u>: אין להוסיף או להוריד מצבים מהאוטומט.

העתיקו את אוטומט המחסנית למחברת הבחינה והשלימו אותו כך שיקבל את השפה L.



/המשך בעמוד הבא/

8. נתונות השפות L_1 , L_2, L_3 מעל האי״ב $\{a,b\}$:

$L_1=\{w\cdot R(w)|\ w\epsilon\{a^nb^m \mid n=m\%2,\ m\geq 0\}\}$

$L_2=\{a^nb^{2n}a^n \mid n\geq 0\}$

$L_3=\{(ab)^n(ba)^m \mid n \geq m\ ,\ m \geq 0\}$

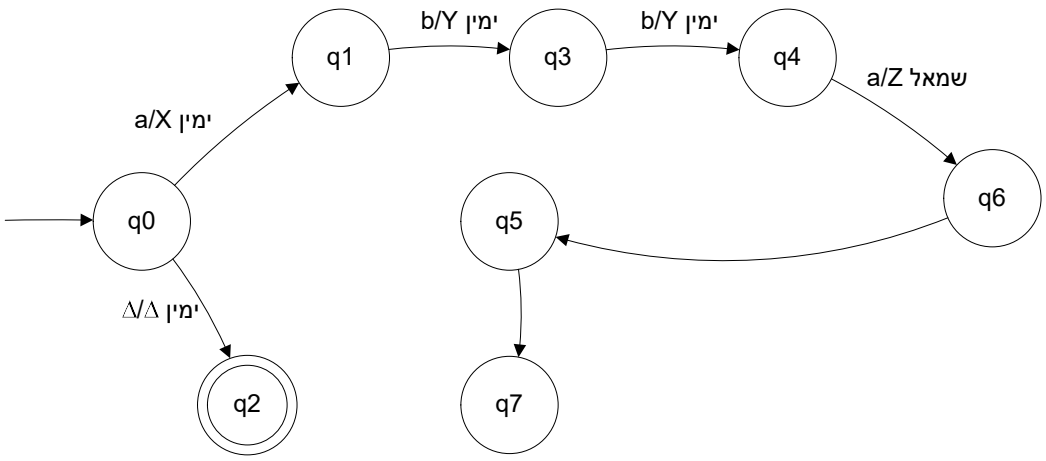
א. (1) כתבו לגבי כל שפה האם היא רגולרית או לא, ונמקו בקצרה.

(2) כתבו מילה שאינה ריקה השייכת לשפת החיתוך של $L_1 \cap L_3$.

ב. לפניכם מכונת טיורינג חלקית שבודקת האם מילה בתחילת הסרט שייכת לשפה L_2 (הנתונה לעיל).
המכונה כוללת את כל המצבים נדרשים (כולל סימון מצב מקבל).
העתיקו למחברת הבחינה את המכונה החלקית הנתונה והשלימו אותה כך שתקבל את השפה L_2 .

הערות:

- אין צורך לשמור את מילת הקלט על הסרט.
- אין להוסיף מצבים.
- ניתן להוסיף מעברים.



/המשך בעמוד הבא/

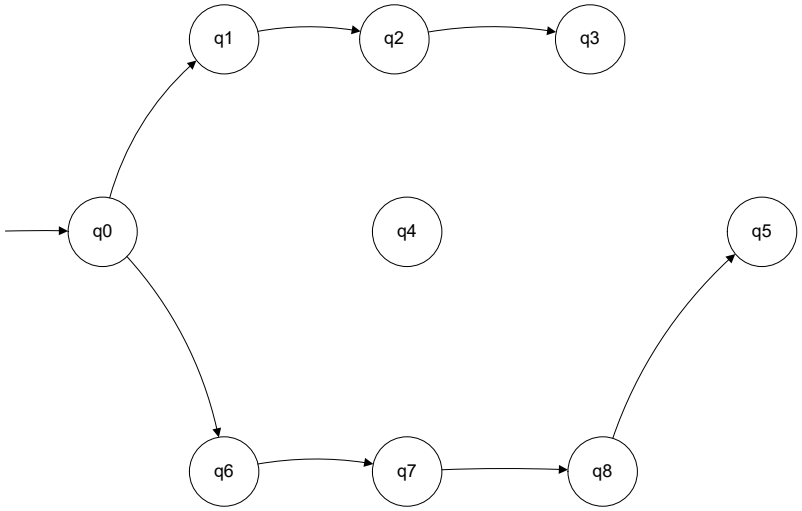
9. לפניכם שני סעיפים, א-ב שאינם קשורים זה לזה. עליכם לענות על שניהם.

א. נתונה השפה L מעל האי״ב {a,b}, שמכילה מלים המקיימות את כל התנאים שלפניכם:

- המילה לא מתחילה ב-ab.
- אם המילה מכילה aa אז היא לא מכילה bb.
- אם המילה מכילה bb אז היא לא מכילה aa.
- המילה חייבת להכיל aa או bb לפחות פעם אחת.

לפניכם אוטומט סופי דטרמיניסטי <u>מלא</u> שמקבל את השפה L. באוטומט חסרים מעברים, סימני קלט ומצבים מקבלים. העתיקו את האוטומט למחברת הבחינה והשלימו אותו כך שיקבל את השפה L.

<u>הערה</u>: אין להוסיף או להוריד מצבים ואין להוריד מעברים מהאוטומט הנתון.



ב. נתונה השפה L מעל האי״ב {a,b,c}:

$$L=\{a^{2m}b^{m-1}c \mid m\geq 1\}$$

(1) מהי המילה הקצרה ביותר בשפה L?

(2) האם השפה L רגולרית? נמקו בקצרה.

/המשך בעמוד הבא/

<u>תכנות מונחה עצמים</u>

10. בשאלה זו שני נוסחים: בשפת Java בעמודים 15-16, ובשפת C# בעמודים 17-18.

<u>לפותרים בשפת Java</u>

לפניכם כותרות המחלקות BB, CC והתכונות שלהן:

```
public class BB
{
   private int a;
   protected int b;

      ****
}

public class CC extends BB
{
   private int c;

      ****
}
```

<u>הערה</u>: שימו לב – לשתי המחלקות <u>אין פעולות get ו- set</u>.

לפניכם המחלקה Tester הכוללת פעולה ראשית:

```
public class Tester{
   public static void main(String[] args)
   {
      BB [] b = new BB[8];
      b[0] = new BB();
      b[1] = new BB(16);
      b[2] = new BB(b[0]);
      b[3] = new BB(b[1]);
      b[4] = new CC(9);
      b[5] = new CC(5);
      b[6] = new CC(b[0]);
      b[7] = new CC(b[1]);
   }
}
```

א. לפניכם תרשים עצמים שנוצרו בעקבות הרצת קטע הקוד.

כתבו במחלקות BB ו- CC את הפעולות הבונות הנדרשות כדי לקבל את העצמים שבתרשים.

ציינו בעבור כל אחד מן העצמים שבתרשים את הפעולה הבונה על פיה הוא נוצר.

<u>הערה</u>: <u>אין</u> להוסיף פעולות שאינן בונות במחלקות BB ו- CC. פתרון הכולל פעולות שאינן בונות לא יזוכה בנקודות.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

0	1	2	3	4	5	6	7
BB	BB	BB	BB	CC	CC	CC	CC
a= 10 b= 1	a= 10 b= 16	a= 10 b= 1	a= 10 b= 16	a= 10 b= 9 c= 7	a= 10 b= 5 c= 7	a= 10 b= 1 c= 19	a= 10 b= 16 c= 19

ב. נתון קטע קוד תקין והפלט שהתקבל בעקבות הרצתו. כמו כן, נתון קטע קוד עבורו התקבלה שגיאת הידור (קומפילציה).

ממשו את הפעולות goo ו- foo במחלקות המתאימות (BB ו- CC) על פי עקרונות תכנות מונחה עצמים, כך שיתקבל הפלט שלפניכם (יש לממש במחלקות רק את הפעולות הנחוצות כדי שהקוד ירוץ).

<u>קטע קוד תקין</u>:

```
b[0].foo(0);          //1
b[0].foo(3);          //2
((CC)b[4]).foo();     //3
((CC)b[6]).foo();     //4
((CC)b[5]).goo();     //5
((CC)b[7]).goo();     //6
```

<u>הפלט שהתקבל עבור הרצתו</u>:

```
BB foo: 11        //1
BB foo: 14        //2
CC foo: 7         //3
BB foo: 26
CC foo: 19        //4
BB foo: 30
CC goo            //5
CC goo            //6
```

כמו כן, שתי הפעולות בקטע הקוד שלפניכם <u>אינן תקינות</u> ומתקבלת שגיאת הידור (קומפילציה).

```
b[3].foo();
b[2].goo();
```

/המשך בעמוד הבא/

<u>לפותרים בשפת C#:</u>

לפניכם כותרות המחלקות CC, BB והתכונות שלהן:

```
public class BB
{
   private int a;
   protected int b;

     ****
}

public class CC: BB
{
   private int c;

     ****
}
```

<u>הערה:</u> שימו לב – לשתי המחלקות <u>אין פעולות Get ו- Set</u>.

לפניכם המחלקה Tester הכוללת פעולה ראשית:

```
public class Tester{
   public static void Main(string[] args)
   {
      BB [] b = new BB[8];
      b[0] = new BB();
      b[1] = new BB(16);
      b[2] = new BB(b[0]);
      b[3] = new BB(b[1]);
      b[4] = new CC(9);
      b[5] = new CC(5);
      b[6] = new CC(b[0]);
      b[7] = new CC(b[1]);
   }
}
```

א. לפניכם תרשים עצמים שנוצרו בעקבות הרצת קטע הקוד.

כתבו במחלקות BB ו־ CC את הפעולות הבונות הנדרשות כדי לקבל את העצמים שבתרשים.

ציינו בעבור כל אחד מן העצמים שבתרשים את הפעולה הבונה על פיה הוא נוצר.

הערה: <u>אין</u> להוסיף פעולות שאינן בונות במחלקות BB ו־ CC. פתרון הכולל פעולות שאינן בונות לא יזוכה בנקודות.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

0	1	2	3	4	5	6	7
BB	BB	BB	BB	CC	CC	CC	CC
a= 10 b= 1	a= 10 b= 16	a= 10 b= 1	a= 10 b= 16	a= 10 b= 9 c= 7	a= 10 b= 5 c= 7	a= 10 b= 1 c= 19	a= 10 b= 16 c= 19

ב. נתון קטע קוד תקין והפלט שהתקבל בעקבות הרצתו. כמו כן, נתון קטע קוד עבורו התקבלה שגיאת הידור (קומפילציה).
ממשו את הפעולות Goo ו- Foo במחלקות המתאימות (BB ו- CC) על פי עקרונות תכנות מונחה עצמים, כך שיתקבל הפלט שלפניכם (יש לממש במחלקות רק את הפעולות הנחוצות כדי שהקוד ירוץ).

<u>קטע קוד תקין</u>:

```
b[0].Foo(0);          //1
b[0].Foo(3);          //2
((CC)b[4]).Foo();     //3
((CC)b[6]).Foo();     //4
((CC)b[5]).Goo();     //5
((CC)b[7]).Goo();     //6
```

<u>הפלט שהתקבל עבור הרצתו</u>:

```
BB Foo: 11        //1
BB Foo: 14        //2
CC Foo: 7         //3
BB Foo: 26
CC Foo: 19        //4
BB Foo: 30
CC Goo            //5
CC Goo            //6
```

כמו כן, שתי הפעולות בקטע הקוד שלפניכם <u>אינן תקינות</u> ומתקבלת שגיאת הידור (קומפילציה).

```
b[3].Foo();
b[2].Goo();
```

/המשך בעמוד הבא/

11. בשאלה זו שני נוסחים: בשפת Java בעמודים 19-20, ובשפת #C בעמודים 21-22.

<u>לפותרים בשפת Java</u>

בחנות משחקים בנו מערכת לניהול המשחקים, ובה המחלקות הבאות:

Game, VRGame, ComputerGame, BoardGame.

להלן פירוט תכונות המחלקות:

- **Game** - משחק

  תכונות המחלקה:

  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם

- **ComputerGame** - משחק מחשב

  תכונות המחלקה:

  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם
  - memory – זיכרון מינימלי, מטיפוס שלם

- **VRGame** - משחק מציאות מדומה

  תכונות המחלקה:

  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם
  - memory – זיכרון מינימלי, מטיפוס שלם
  - needVR – האם נדרשים משקפים, מטיפוס בוליאני
  - age – גיל מינימלי לשחקן במשחק, מטיפוס שלם

- **BoardGame** – משחק לוח

  תכונות המחלקה:

  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם
  - players – מספר שחקנים מקסימלי במשחק, מטיפוס שלם

א.

(1) שרטטו תרשים היררכיה המתאר את הקשר בין המחלקות של המערכת הממוחשבת.
יש לסמן ירושה באמצעות החץ ⟵ .

(2) כתבו את כותרות המחלקות ואת התכונות שלהן על פי עקרונות תכנות מונחה עצמים. הניחו שהפעולות get ו- set קיימות לכל התכונות של המחלקות, ואין צורך למשש אותן.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

נתונה כותרת הפעולה הבונה של המחלקה Game:

public Game(int price, int code)

אין צורך לממש את הפעולה. דירוג המשחק מאותחל לערך 0.

(3) לפניכם כותרת הפעולה הבונה של המחלקה ComputerGame. הפעולה מקבלת מחיר price, קוד code, וזיכרון מינימלי memory.

public ComputerGame (int price, int code, int memory)

ממשו את הפעולה הבונה.

ב. נתונה המחלקה Store – חנות, ולה שתי תכונות:

- games – מערך של משחקים, מטיפוס Game בגודל 1000.
- count – כמות המשחקים במערך.

המשחקים נשמרים מתחילת המערך ברצף, אך ייתכן שהמערך אינו מלא עד הסוף. כמו כן, המערך אינו ממויין.

כתבו פעולה פנימית ששמה avgRate במחלקה Store המקבלת מספר המייצג סוג המשחק:
המספר 1 – מייצג את המשחק VRGame.
המספר 2 – מייצג את המשחק ComputerGame.
הפעולה מחזירה מספר ממשי המייצג את ממוצע הדירוגים של כל המשחקים מאותו סוג בחנות.
הניחו שקיים במערך לפחות משחק אחד מכל סוג.

/המשך בעמוד הבא/

<u>לפותרים בשפת C#</u>

בחנות משחקים בנו מערכת לניהול המשחקים, ובה המחלקות הבאות:
Game, VRGame, ComputerGame, BoardGame.
להלן פירוט תכונות המחלקות:

- **Game** - משחק
  תכונות המחלקה:
  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם
- **ComputerGame** - משחק מחשב
  תכונות המחלקה:
  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם
  - memory – זיכרון מינימלי, מטיפוס שלם
- **VRGame** - משחק מציאות מדומה
  תכונות המחלקה:
  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם
  - memory – זיכרון מינימלי, מטיפוס שלם
  - needVR – האם נדרשים משקפים, מטיפוס בוליאני
  - age – גיל מינימלי לשחקן במשחק, מטיפוס שלם
- **BoardGame** – משחק לוח
  תכונות המחלקה:
  - rate – דירוג, מטיפוס מספר שלם
  - code – קוד משחק, מטיפוס שלם
  - price – מחיר משחק, מטיפוס שלם
  - players – מספר שחקנים מקסימלי במשחק, מטיפוס שלם

א.

(1) שרטטו תרשים היררכיה המתאר את הקשר בין המחלקות של המערכת הממוחשבת.
יש לסמן ירושה באמצעות החץ ⟵ .

(2) כתבו את כותרות המחלקות ואת התכונות שלהן על פי עקרונות תכנות מונחה עצמים. הניחו שהפעולות Get ו- Set קיימות לכל התכונות של המחלקות, ואין צורך למשש אותן.

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

נתונה כותרת הפעולה הבונה של המחלקה Game:

public Game(int price, int code)

אין צורך לממש את הפעולה. דירוג המשחק מאותחל לערך 0.

(3) לפניכם כותרת הפעולה הבונה של המחלקה ComputerGame. הפעולה מקבלת מחיר price, קוד code, וזיכרון מינימלי memory.

public ComputerGame (int price, int code, int memory)

ממשו את הפעולה הבונה.

ב. נתונה המחלקה Store – חנות, ולה שתי תכונות:

- games – מערך של משחקים, מטיפוס Game בגודל 1000.
- count – כמות המשחקים במערך.

המשחקים נשמרים מתחילת המערך ברצף, אך ייתכן שהמערך אינו מלא עד הסוף. כמו כן, המערך אינו ממויין.

כתבו פעולה פנימית ששמה AvgRate במחלקה Store המקבלת מספר המייצג סוג המשחק:
המספר 1 – מייצג את המשחק VRGame.
המספר 2 – מייצג את המשחק ComputerGame.
הפעולה מחזירה מספר ממשי המייצג את ממוצע הדירוגים של כל המשחקים מאותו סוג בחנות.
הניחו שקיים במערך לפחות משחק אחד מכל סוג.

/המשך בעמוד הבא/

12. בשאלה זו שני נוסחים: בשפת Java בעמודים 23-24, ובשפת C# בעמודים 25-26.

<u>לפותרים בשפת Java</u>

לפניכם המחלקות E, F, G, H:

```
public class E {
   protected int x;
   public E() {
      this.x = 5;
      System.out.println("E x = " + this.x);
   }
   public E(int x) {
      this.x = x;
   }
   public int calc() {
      return this.x;
   }
}
```

```
public class F extends E {
   private static int count = 0;
   public F() {
      super(8);
      count++;
   }
   public F(int x) {
      super(x);
      count++;
      System.out.println("F count = " + count);
   }
   public int calc() {
      return this.x + 10;
   }
}
```

```
public class G extends F {
   public G() {
      super(3);
      this.x =this.x* 2;
      System.out.println("G x = " + this.x);
   }
   public int calc() {
      return super.calc() + 1;
   }
   public int squre() {
      return this.x*this.x;
   }
}
```

```
public class H extends E {
   public H() {
      super();
      this.x = this.x + 5;
      System.out.println("H x = " + this.x);
   }
   public int calc() {
      return this.x/2;
   }
}
```

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

נתונה המחלקה Tester:

```
public class Tester
{
  public static void main(String[] args)
  {
    E e1 = new E();
    E e2 = new F();
    F f1 = new G();
    E e3 = new H();
    E a = new G();
    E b = new H();
    F c = new G();
    // ***
  }
}
```

א. הציגו את העצמים שנוצרו בפעולה main (סוג העצם, סוג ההפנייה וערכי התכונות), 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 המתקבל תקין או לא תקין.
אם הקוד תקין – כתבו את הפלט. אם הקוד אינו תקין, הסבירו מדוע.

<u>שימו לב</u>: אין קשר בין הפקודות הבאות. הפקודות אינן תלויות זו בזו.

1. `F f = new E();`
2. `G g = new F();`
3. `F f2 = (F)a;`
4. `G g2 = (G)c;`
5. `F f3 = (F)b;`
6. `System.out.println(((G)a).squre());`
7. `System.out.println(a.squre());`
8. `System.out.println(((H)b).calc());`
9. `System.out.println(((G)b).squre());`
10. `System.out.println(f1.squre());`

/המשך בעמוד הבא/

<u>לפותרים בשפת C#</u>

לפניכם המחלקות E, F, G, H:

```
public class E {
   protected int x;
   public E(){
      this.x = 5;
      Console.WriteLine(״E x = ״ + this.x);
   }
   public E(int x) {
      this.x = x;
   }
   public int Calc() {
      return this.x;
   }
}
-------------------------------------------------
public class F: E {
   private static int count = 0;
   public F(): base(8){
      count++;
   }
   public F(int x): base(x) {
      count++;
      Console.WriteLine(״F count = ״ + count);
   }
   public int Calc() {
      return this.x + 10;
   }
}
```

```
public class G: F {
   public G(): base(3) {
      this.x = this.x * 2;
      Console.WriteLine(״G x = ״ + this.x);
   }
   public int Calc() {
      return base.Calc() + 1;
   }
   public int Squre() {
      return this.x*this.x;
   }
}
-------------------------------------------------
public class H: E {
   public H(): base(){
      this.x = this.x + 5;
      Console.WriteLine (״H x = ״ + this.x);
   }
   public int Calc() {
      return this.x/2;
   }
}
```

(שימו לב: המשך השאלה בעמוד הבא.)

/המשך בעמוד הבא/

נתונה המחלקה Tester:

```
public class Tester
{
  public static void Main(string[] args)
  {
    E e1 = new E();
    E e2 = new F();
    F f1 = new G();
    E e3 = new H();
    E a = new G();
    E b = new H();
    F c = new G();

    // ***
  }
}
```

א. הציגו את העצמים שנוצרו בפעולה Main (סוג העצם, סוג ההפנייה וערכי התכונות), 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 המתקבל תקין או לא תקין.
אם הקוד תקין – כתבו את הפלט. אם הקוד אינו תקין, הסבירו מדוע.

<u>שימו לב</u>: אין קשר בין הפקודות הבאות. הפקודות אינן תלויות זו בזו.

1. `F f = new E();`
2. `G g = new F();`
3. `F f2 = (F)a;`
4. `G g2 = (G)c;`
5. `F f3 = (F)b;`
6. `Console.WriteLine(((G)a).Squre());`
7. `Console.WriteLine(a.Squre());`
8. `Console.WriteLine(((H)b).Calc());`
9. `Console.WriteLine(((G)b).Squre());`
10. `Console.WriteLine(f1.Squre());`

בהצלחה!