מדעי המחשב ב'

פתרון בחינת הבגרות

פרק א - עיצוב תכנה

שאלה 1:

list1: 2 → 4 → 5 → 1 → 1 → 9 → null

list2: 2 → 4 → 5 → 1 → 4 → null

sod1

node1 מצביע על	node2 מצביע על	i	i ≤ 4	node1 == null || node2 == null	i == 1 || i == 4	node1.getInfo() ≠ node2.getInfo()	ערך מוחזר
2	2	1	כן	לא	כן	לא	
4		2	כן	לא	לא		
5		3	כן	לא	לא		
1		4	כן	לא	כן	כן	
		5	לא				false

(הפעולה מחזירה אמת אם האיבר הראשון והרביעי ב- list2 שווים לאיבר הראשון ב- list1, ושקר אחרת.
אם אין 4 איברים ב- list1 או list2 רשימה ריקה, יוחזר שקר).

סיבוכיות הפעולה: O(1) הפעולה מבצעת בדיוק 4 צעדים (קבוע) וכל הפעולות הפנימיות ב- O(1)

sod2

node1 מצביע על	node2 מצביע על	node1 != null && node2 != null	node1.getInfo() ≠ node2.getInfo()	ערך מוחזר
2	2	כן	לא	
4	4	כן	לא	
5	5	כן	לא	
1	1	כן	לא	
1	4	כן	כן	false

(הפעולה מחזירה אמת אחת הרשימות מוכלת באחרת בשלמותה, ושקר אחרת). (מוכלת = חלקית או לחלוטין)

סיבוכיות הפעולה: O(n) הפעולה עוברת על n האיברים הראשונים בשתי הרשימות במקביל,
וכל הפעולות הפנימיות ב- O(1).

sod3

node1 מצביע על	node1 ≠ null	found	node2 מצביע על	node2 != null && ! found	node1.getInfo() == node2.getInfo()	! found	ערך מוחזר
2	כן	~~F~~ T	~~2~~ 4	~~כן~~ לא	כן	F	
4	כן	~~F~~ T	2 4 5	כן כן לא	לא כן	F	
5	כן	~~F~~ T	2 4 5 1	כן כן כן לא	לא לא כן	F	
1	כן	~~F~~ T	2 4 5 1 4	כן כן כן כן	לא לא לא כן	F	
1	כן	~~F~~ T	2 4 5 1 4	כן כן כן כן	לא לא לא כן	F	
9	כן	F	2 4 5 1 4 null	כן כן כן כן כן לא	לא לא לא לא לא	T	false

(הפעולה מחזירה אמת אם כל האיברים ב- list1 נמצאים ב- list2 ושקר אחרת).

סיבוכיות הפעולה: $O(n^2)$ עבור כל אחד מהאיברים ב- list1 עוברת הפעולה על כל האיברים ב- list2.
כל הפעולות הפנימיות הן ב- $O(1)$

שאלה 2: Java:

```
//------------- סעיף א --------------------
//--- פעולה המחזירה את הערך הגדול ביותר בעץ טרינארי ---
//---                                  אם העץ ריק, יוחזר -1 ---
public static int big (TriTreeNode<Integer> t)
{
      if (t == null)
            return -1;
      int x = Math.max(big(t.getLeft()), big(t.getMiddle()));
      int y = Math.max(x, big(t.getRight()));
      return Math.max(y, t.getInfo());
}
```

```
//------------- סעיף ב --------------------
//--- פעולה המחזירה אמת אם לכל צמתי העץ ---
//---          לכל היותר 2 בנים, ושקר אחרת ---
public static boolean noThree (TriTreeNode<Integer> tr)
{
      if (tr == null) return true;
      if (numOfSons(tr) == 3)
            return false;
      return      noThree (tr.getLeft())   &&
                  noThree (tr.getMiddle()) &&
                  noThree (tr.getRight());
}

//--- פעולה המחזירה את מספר הבנים של צומת ---
public static int numOfSons (TriTreeNode<Integer> tr)
{
      if (tr == null) return 0;
      int count = 0;
      if (tr.getLeft()   != null) count ++;
      if (tr.getMiddle() != null) count ++;
      if (tr.getRight()  != null) count ++;
      return count;
}
```

C# : נכתב ע"י **ראמי ג'באלי**

```
//--- פעולה המחזירה את האיבר המקסימאלי מבין 4 ספרים ---
public static int Max4(int a, int b, int c,int d)
{
      return Math.Max(Math.Max(a, b), Math.Max(c, d));
}

//--- פעולה המחזירה את האיבר המקסימאלי בעץ ---
public static int Big(TrItReeNode<int> t)
{
      if (t == null)
            return -1;
      int maxLeft = Big(t.GetLeft());
      int maxmiddle = Big(t.GetMiddle());
      int maxRight = Big(t.GetRight());
      return Max4(t.GetInfo(),maxLeft,maxmiddle,maxRight);
}

//--- פעולה המחזירה אמת אם בכל צומת יש לכל היותר שני בנים, ושקר אחרת ---
public static bool NoThree(TrItReeNode<int> tr)
{
      if (tr == null)
            return true;
      if (tr.GetLeft() != null &&
          tr.GetRight() != null &&
          tr.GetMiddle() != null)
                return false;
      return NoThree(tr.GetLeft()) &&
             NoThree(tr.GetMiddle()) &&
             NoThree(tr.GetRight()) ;
}
```

שאלה 3:

Java :

א.

```
/*    BiStone - המחלקה: אבן משחק */
public class BiStone
{
      private int ones;      //    מספר חד ספרתי
      private int tens;      //    מספר דו ספרתי

      //--- ones פעולה בונה המקבלת מספר חד ספרתי תקין ---
      //--- tens ומספר דו ספרתי תקין ---
      public BiStone(int ones, int tens)
      {
            this.ones = ones;
            this.tens = tens;
      }

      //--- ones פעולה המחזירה אמת אם ---
      //--- tens שווה לספרת האחדות של ---
      //---                    אחרת ושקר ---
      public boolean sameDigit()
      {
            return this.ones == this.tens%10;
      }

}

/*    Stones - המחלקה: אוסף אבני המשחק  */
public class Stones
{
      private List<BiStone> lst;
      public static final int MAX = 7;

      public Stones()
      {
            this.lst = new List<BiStone>();
            for (int i = 0 ; i < MAX ; i++)
                  for (int j = 0 ; j < MAX ; j++)
                  {
                        BiStone bs = new BiStone (i, j+10);
                        lst.insert(null, bs);
                  }
      }

}
```

באמצעות רשימה:

הערה: השימוש בקבועים אינו חובה בבחינה.
הקבועים מאפשרים גמישות בשינוי ממדי המשחק.

```
/*   - Stonesהמחלקה: אוסף אבני המשחק */
public class Stones
{
      public static final int MAX = 7;
      public static final int MAX_SIZE = MAX * MAX;
      private BiStone [] arr;

      public Stones()
      {
            this.arr = new BiStone[MAX_SIZE];
            int p = 0;
            for (int i = 0 ; i < MAX ; i++)
                  for (int j = 0 ; j < MAX ; j++)
                  {
                        BiStone bs = new BiStone (i, j+10);
                        arr[p] = bs;
                        p ++;
                  }
      }
}
```

באמצעות מערך:

ב.

```
//---    num ומספר lst פעולה המקבלת כפרמטר רשימה ---
//---          ומחזירה רשימה חדשה כפי שמתואר בשאלה ---
public static List<Integer> listBuild(List<Integer> lst, int num)
{
      List<Integer> lst1 = new List<Integer>();

      for (int i = 1; i < num; i++)
            if (! exist (lst, i))
                  lst1.insert(null, i);
      return lst1;
}

//---  x פעולה המחזירה אמת אם המספר ---
//---       נמצא ברשימה, ושקר אחרת ---
public static boolean exist(List<Integer> lst, int x)
{
      Node<Integer> pos = lst.getFirst();
      while (pos != null)
      {
            if (pos.getInfo() == x)
                  return true;
            pos = pos.getNext();
      }
      return false;
}
```

הפעולה מבצעת num פעמים את הפעולה exist שסיבוכיותה $O(n)$.

עבור num גדול מאוד (ערכו מתקרב ל- n), ניתן להתייחס לסיבוכיות כאל $O(n^2)$

עבור num קטן מאוד, ניתן להתייחס אליו כאל קבוע, ואז הסיבוכיות תהיה $O(n)$

C# : נכתב ע"י **ראמי ג'באלי**

א.

```
using . . .
public class BiStone
{
      private int x, y;

      public BiStone(int x, int y)
      {
            this.x = x;
            this.y = y;
      }
      public bool Check()
      {
            return x == y % 10;
      }
}

using . . .
public class Stones
{
      private BiStone[] arr;
      private int k=0;

      public Stones()
      {
            this.arr = new BiStone[49];
            for(int i=0;i<=6;i++)
                  for(int j=10;j<=16;j++)
                  {
                        arr[k]=new BiStone(i,j);
                        k++;
                  }
      }
}
```

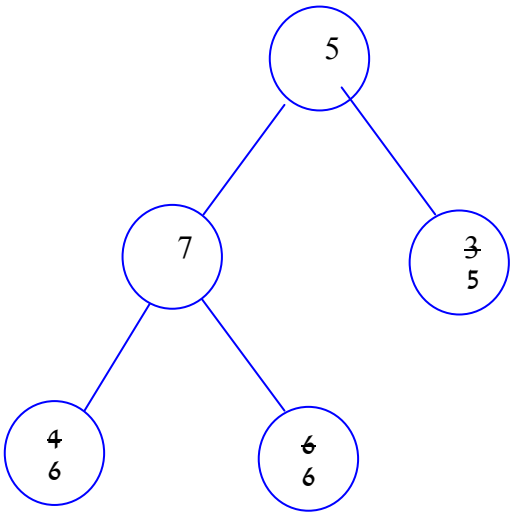
ב.

```
//---------- פתרון שאלה 3 - סעיף ב' - ע"י רמי ג'באלי -----
//---    פעולה המקבלת כפרמטר רשימה lst ומספר num  ---
//---          ומחזירה רשימה חדשה כפי שמתואר בשאלה  ---
public static List<int> Build(List<int> lst, int num)
{
      List<int> list = new List<int>();
      Node<int> pos = list.GetFirst();

      for (int i = 1 ; i < num ; i++)
            if (Found(lst, i) == false)
                  pos = list.Insert(pos, i);

      return list;
}

//---  פעולה המחזירה אמת אם המספר x  ---
//---       נמצא ברשימה, ושקר אחרת  ---
public static bool Found(List<int> list, int x)
{
      Node<int> pos = list.GetFirst();
      while (pos != null)
      {
            if (pos.GetInfo() == x)
                  return true;
            pos = pos.GetNext();
      }

      return false;
}
```

Java :

שאלה 4:

א.



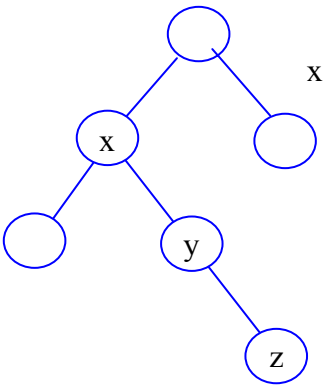
(i) הפעולה amir(tree, 4) מציבה בכל עלה ערך השווה ל- x + גובה/רמת העלה.

(ii) הפעולה amir (tree, 0) תציב בכל עלה את גובה/רמת העלה.

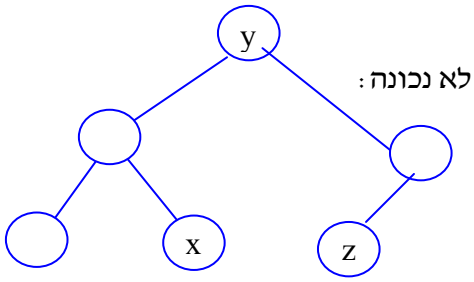
ב. עץ חיפוש בינארי:

(1) תוצאת סריקה בסדר תוכי: ... , x , y , z , ...
האם יתכן ש- x == y ??
הטענה נכונה רק בחלק מהמקרים.

נכונה כאשר x = y = z



לא נכונה:



(ii) x ו- y הם ערכי שני עלים, כך ש- x נמצא משמאל ל- y
אז הטענה כי x > y אינה נכונה תמיד.
סריקת עץ חיפוש בינארי תציג תמיד את הנתונים ממויינים בדר עולה, ולכן הערך שבצד שמאל יהיה תמיד קטן או שווה לערך שבצד ימין ולעולם לא גדול יותר.

פרק ב'

מערכות מחשב ואסמבלר

הפתרון לפרק זה נכתב ע"י: **רונית מרציאנו**,
מתיכון עירוני ב', מודיעין

תרגיל 5:

א. טבלת מעקב (כל הטבלה בהקסה)
לא חייבים לשים תיאור של אוגר הדגלים

										FLAGS					
NIBBLE	M	NUMBER	AX		BX		CX		DX		CF	ZF	OF	SF	הפקודה
			AH	AL	BH	BL	CH	CL	DH	DL					
09h	0F000h	0E539h					00h	04h	00h	00h	0	1	0	0	xor dx,dx
									09h						
							00h	0Dh	**90h**		0	0	1	1	shl dh,cl
			E5h	39h											
					F0h	00h									mov bx,M
					E0h	00h					0	0	0	1	and bx,ax
					70h	00h					0	0	0	0	xor bx,dx
			CAh	72h							1	0	0	1	shl ax,1
					F0h	00h	00h	0Ch							mov bx,M
					C0h	00h					0	0	0	1	and bx,ax
					50h	00h					0	0	0	0	xor bx,dx
			94h	E4h							1	0	0	1	shl ax,1
					F0h	00h	00h	0Bh							mov bx,M
					90h	00h					0	0	0	1	and bx,ax
					00h	**00h**					0	**1**	0	0	xor bx,dx
							01h								jz ok

ב. מה מבצע הקטע?

הקטע בודק אם המשתנה NIBBLE נמצא במשתנה NUMBER,
אם כן יוכנס לאוגר CH הערך 1, אחרת יוכנס לאוגר CH הערך 0.

תרגיל 6:

א. טבלת מעקב (כל הטבלה בהקסה)

					FLAGS				
K	AX		CX		CF	ZF	OF	SF	הפקודה
	AH	AL	CH	CL					
5Eh			00h	02h					
	5Eh	5Eh			0	0	0	0	
	79h	78h			1	0	1	0	shl ax,cl
			00h	01h	1	0	0	0	dec cl
	3Ch	78h							shr ah,cl
	4Bh	78h			0	0	1	1	add ah,al
	3Ch	78h			0	0	1	0	sub ah,al
	01h	78h	00h	00h					mov ah,1

ב. I מבצע את הנדרש

II אינו מבצע את הנדרש.

הקטע יבצע את הנדרש רק עם הערך הנמצא ב AX קטן מ H1000

דוגמא עבור הערך AX = 0352H

הקטע יבצע את הנדרש

עבור הערך AX = 45A3H

בסוף הקטע AX = 45A0H

III אינו מבצע את הנדרש.

בדיוק אותה הבעיה כמו בקטע הקודם עובד רק עם ערכים קטנים מ H1000

דוגמא: עבור הערך AX= 3A1CH

בסוף הקטע AX = 0AICH

IV מבצע את הנדרש

תרגיל 7:

א.
```
while ((a<b) || (a>0))
    a = a-1;
```

```
        MOV   AX,A
        MOV   BX,B
NEXT:   CMP   AX,BX
        JGE   NOTSMALL
        DEC   AX
        JMP   NEXT
NOTSMALL: CMP AX,0
        JL    SOF
        DEC   AX
        JMP   NEXT
SOF:    NOP
```

ב.
```
AND   AL,00111100B    (1)
OR    AL,11000011B    (2)
OR    AL,00001111B    (3)
```

(4)
- כמה תשובות אפשריות:
  - `MOV AX,A+8`
  - `MOV AX,A[8]`
- כמה תשובות אפשריות:
  - `LEA AX,A+8`
  - `LEA AX,A[8]`

שאלה 8:

```
DATA  SEGMENT
        ARR     DW              100 DUP(0)
        MIN     DW              ?
        NUM   DW                0
ENDS

STACK SEGMENT
        DW   128  dup(0)
ENDS

CODE SEGMENT:
START:
        MOV     AX,DATA
        MOV     DS,AX
        MOV     ES,AX
        ;       my program
        MOV     SI, 2           ;   index arr
        MOV     CX, 99          ;   loop counter
        MOV     BX, 1           ;   min counter
        MOV     DX,  ARR[0]     ;   keep small value
NEXT:   CMP     DX, ARR[SI]
        JB      CONT
        JE      EQUL
        MOV     DX, ARR[SI]     ;   find smaller then small
        MOV     BX, 1           ;   start again small counter
        JMP     CONT
EQUL:   INC     BX
CONT:   INC     SI              ;   mov 2 bytes
        INC     SI
        LOOP    NEXT
        MOV     NUM, BX
        MOV     MIN ,DX
        MOV     AX, 4C00h       ;   exit to operating system.
        INT 21h
ENDS

END   START         ;   set entry point and stop the assembler.
```

פרק ב'

מבוא לחקר ביצועים

שאלה 9:

שאלה 10:

שאלה 11:

שאלה 12:

פרק ב'

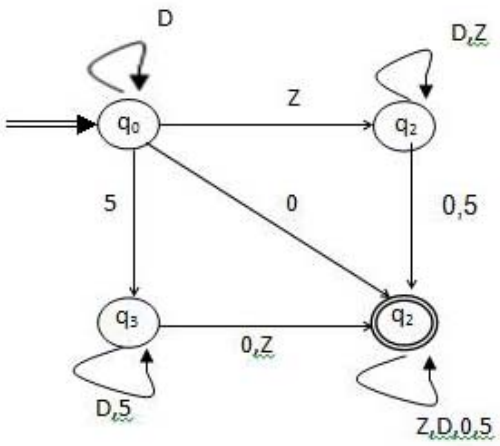
מודלים חישוביים

הפתרון לפרק זה נכתב ע"י **רחל לודמר**.

תרגיל 13:

13.

א. שים לב, המילה תתקבל ע"י האוטומט:
אם אחת מספרות המספר היא 0 או המספר מכיל ספרה 5 וספרה זוגית (קבוצה Z)
ולכן מכפלת הספרות מתחלקת 10:



ב.

1. (i) 000 - לא מתקבלת.

(ii) 01000 - מתקבלת.

$$q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_3 \xrightarrow{0} q_3 \xrightarrow{0} q_3$$

(iii) 1100 - לא מתקבלת.

(iv) 00101 - מתקבלת.

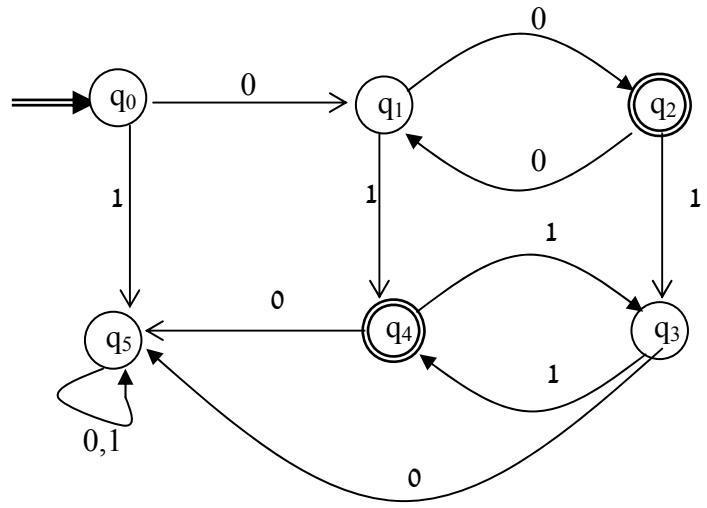
$$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_4 \xrightarrow{1} q_5$$

(v) 1011 - לא מקבלת.

2. השפה L מקבלת מילים מעל הא"ב $\{0,1\}$ המתחילות ב- 01 או נגמרות ב- 01.

שאלה 14:

א.



ב.

(i) הטענה אינה נכונה. המילה aab מתקבלת , מכילה ab ומסתיימת ב- b.

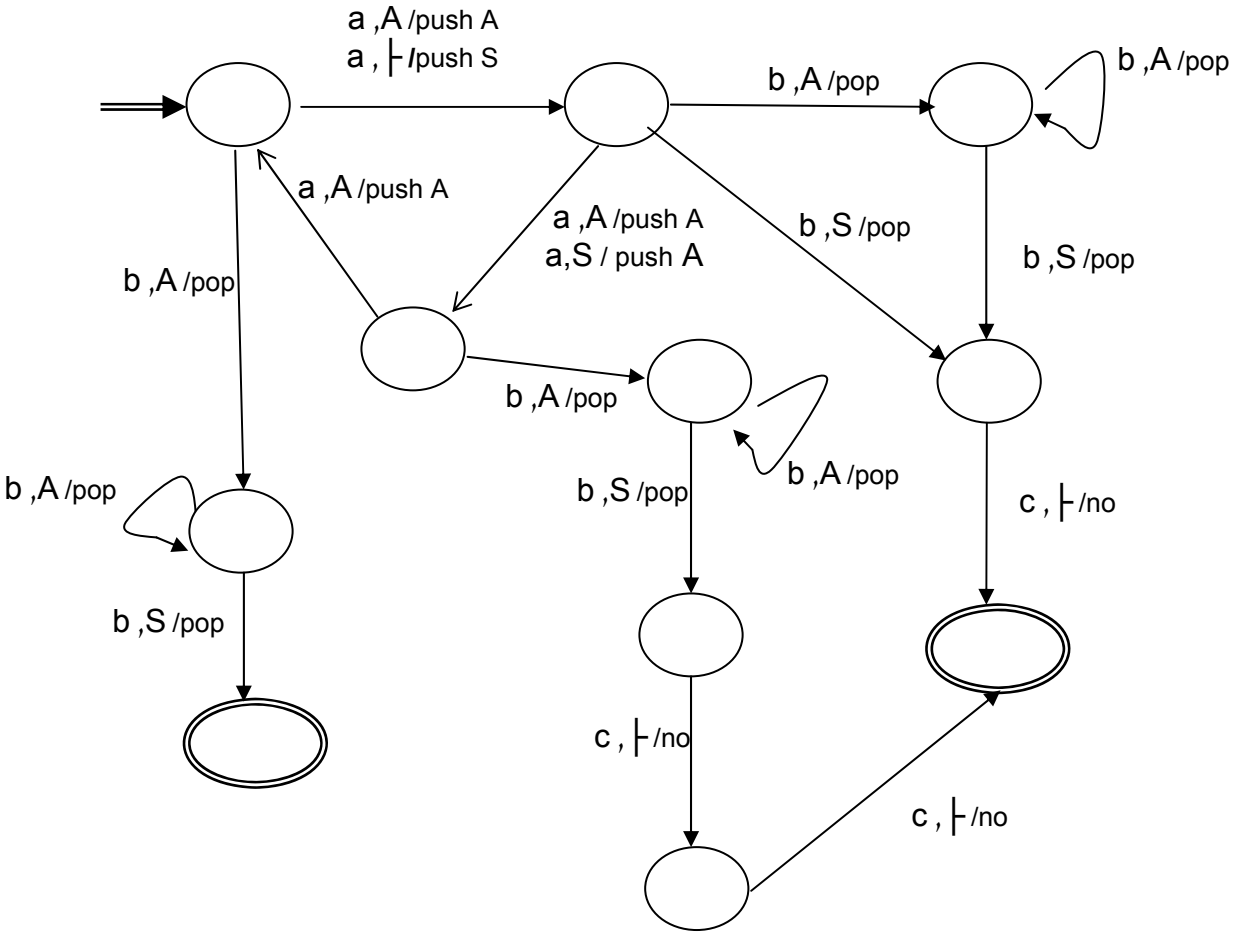
(ii) הטענה לא נכונה. המילה aaaa מתקבלת, לא מכילה ab ואינה מסתיימת ב- b.

(iii) הטענה נכונה. (לא היה צריך לנמק)

ניתן לייצג את השפה L באופן הבא: $L = \{b^n a^m b^k a^t b^r \mid n,m,k,t,r \geq 0\}$

השפה מכילה את כל המילים המסתיימות ב - b עבור: ($n>0$, $m = k = t = r = 0$,
או $n >= 0$, $m > 0$, $k > 0$, $t >= 0$, $r >= 0$
או $n >= 0$, $m > 0$, $k > 0$, $t > 0$, $r > 0$)
או אינן מכילות ab ($t = 0$, $r = 0$, $k = 0$, $m >= 0$, $n >= 0$).

שאלה 15:


a ,A /push A
a , ├ /push S
b ,A /pop
b ,A /pop
a ,A /push A
a ,A /push A
a,S / push A
b ,S /pop
b ,S /pop
b ,A /pop
b ,A /pop
b ,S /pop
b ,A /pop
c , ├ /no
b ,S /pop
c , ├ /no
c , ├ /no

שאלה 16:

א. המילה היא $w = ba^i$

$$w_1 \cdot w = a^i b \cdot ba^i \in L_1$$
$$w_2 \cdot w = a^j b \cdot ba^i \notin L_1 \mid i \neq j$$

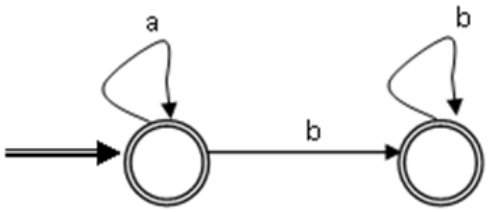
השרשור הוא לא המילה וההופכית שלה

ב. המילה היא $w = b^i c^i$

$$w_1 \cdot w = a^i \cdot b^i c^i \in L_1 \mid \max(i,i) = i \,, i \geq 1$$
$$w_2 \cdot w = a^j \cdot b^i c^i \notin L_1 \mid j > i, \max(j,i) = j$$

ג.

1.



2.

$$L_1 \cap L_2 = \{a^n b^k \mid n,k \geq 0\} \cap (\{a^m cb^r \mid (m+r)\%2 = 1\} \cup \{a^t b^t \mid t \geq 0\}) =$$
$$= \{a^n b^n \mid n \geq 0\}$$

החיתוך אינו רגולרי. קיימת תלות של מניה אינסופית בין הרצפים של a לבין הרצפים של b.

3. מדוע L_2 אינה רגולרית? נכון יש בה חלק שהוא אי רגולרי שבו קיימת תלות של מניה אינסופית בין הרצפים של a לבין הרצפים של b, אבל בקשו להיעזר בתוצאה של סעיף 2.

נסתכל על המשלים של החיתוך, וניעזר בחוקי דמורגן, ובתכונות סגירות.

$$\overline{L_1 \cap L_2} \quad = \quad \overline{L_1} \cup \overline{L_2}$$

בגלל שויון האגפים , מחייב שהשפה $\overline{L_2}$ היא לא רגולרית. אם נניח שהיא רגולרית אז איחוד של שפות רגולריות הוא רגולרי, וזה בסתירה לעובדה שאגף שמאל (החיתוך) הוא אי רגולרי. כעת, אם $\overline{L_2}$ לא רגולרי , גם L_2 לא רגולרי.	L_1 רגולרית. הוכחנו בסעיף 1. לכן מתכונות סגירות של שפות רגולריות, $\overline{L_1}$ רגולרית	$L_1 \cap L_2$ הוא אי רגולרי גם המשלים שלו הוא אי רגולרי (כל המילים שאין בהם רצף a שווה לרצף b)

פרק ב'

תכנות מונחה עצמים Java

הפתרון לפרק זה נכתב ע"י אביטל EVI גרינולד

תרגיל 17:

א. (1) המנגנון המאפשר שתי פעולות בשם זהה באותה מחלקה נקרא: העמסה, overloading.

(2) אפשר להוסיף לממשק המחלקה Node פעולה בונה ריקה:

```
public Node()
{
        this.x = null;
        this.next = null;
}
```

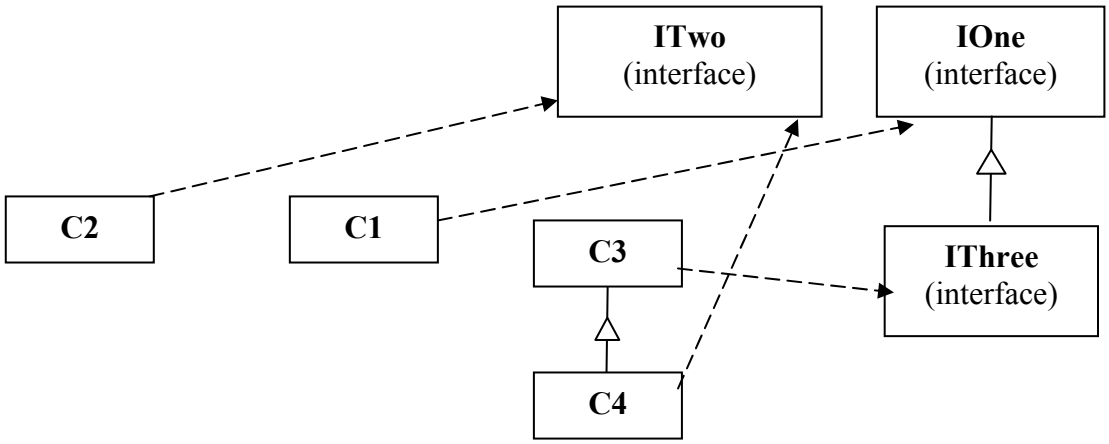
ב.

```
public class Test
{
        public static void main (String [] args)
        {
            Node<Integer>  nd = new Node<Integer>(5); ..... (i)
            nd = new Node1<Integer> (4,nd);..... (ii)
            nd = new Node2<Integer> (3,nd);..... (iii)
            nd = new Node1<Integer> (2,nd);..... (iv)
            nd = new Node<Integer> (1,nd);..... (v)

            while (nd != null)
            {
                System.out.print (nd);
                nd = nd.getNext();
            }
        }
}
```

תרגיל 18:

מיומנויות נבדקות: ממשקים, ירושה של ממשקים ושל מחלקות, מאתרת שגיאות הידור וריצה.

מבנה היררכית המחלקות והממשקים



א.

(i)	ITwo a = **new** ITwo();	**שגיאת הידור**. אי אפשר ליצור עצם ממשק
(ii)	ITwo b = **new** C2();	**תקין**. C2 מתפקד כ- ITwo . ניתן לבצע המרה כלפי מעלה לטיפוס הממשק.
(iii)	C3 c = **new** C4();	**תקין**. C4 סוג של C3 . ניתן לבצע המרה כלפי מעלה לטיפוס מחלקת העל.
(iv)	C2 d = **new** C4();	**שגיאת הידור**. חוסר התאמה בטיפוסים. C4 אינו סוג של C2
(v)	C4 e = **new** C3();	**שגיאת הידור**. חוסר התאמה בטיפוסים. C3 אינו סוג של C4
(vi)	C4 f = (C4)(**new** C3());	**שגיאת ריצה**. אי אפשר לבצע המרה כלפי מטה מטיפוס C3 לטיפוס C4 . המרות מתבצעות בזמן ריצה.
(vii)	IOne g1 = **new** C1(); C4 g2 = **new** C4(); g1=g2;	**תקין**. C1 מתפקד כ IOne . ניתן לבצע המרה כלפי מעלה לטיפוס הממשק. g2 הוא עצם מטיפוס C4 ומציבים אותו למשתנה מאותו הטיפוס. ניתן להסתכל על g2 שהוא עצם מטיפוס C4 כעצם מטיפוס הממשק IOne כי הוא ממש אותו.
(viii)	IOne h1=**new** C4(); ITwo h2=**new** C2(); h2 = h1;	**שגיאת הידור**. שורה ראשונה נכונה כי C4 יורש מ C3 שממש את IThree אשר יורש מ IOne ולכן ממש גם אותו. שורה שנייה נכונה כי C2 ממש את ITwo אבל מאחר ואין קשר בין הממשקים IOne ו- ITwo השורה השלישית תהווה שגיאת הידור - אין התאמה בין הטיפוסים.

ב.

(i) המחלקה B אינה עוברת קומפילציה.
במחלקה A קיימת פעולה בונה עם פרמטר ולכן הפעולה הבונה הריקה כברירת מחדל לא קיימת יותר.

תיקון אפשרי 1: נזמן בפעולה הבונה של B את הפעולה הבונה של מחלקת העל עם הפרמטר.
תיקון אפשרי 2: נוסיף למחלקה A פעולה בונה ריקה.

(ii) כדי לקבל פלט: 10,6 משמאל לימין נבצע את התיקון הבא:

- מחלקה A ללא שינוי
- בפעולה הבונה ל המחלקה :B
  נחליף את השורה הראשונה
  בזימון של הבנאי של מחלקת העל עם הערך 10.

```
public class B extends A
{
   public B (int k)
   {
        super (10);
        A a=new A(10);
        System.out.println (k + " ");
   }
}
```

או:

- מחלקה B נשארת ללא שינוי.
- הוספת פעולה בונה ללא פרמטרים למחלקה A.

```
public class A
{
    public A()   {  }
    public A (int k)
    {
        System.out.print(k+" ");
    }
}
```

```
public class B extends A
{
        public B (int k)
        {
                A a=new A(10);
                System.out.println (k + " ");
        }
}
```

(iii) כדי לקבל את הפלט משמאל לימין: 6 10 6 :

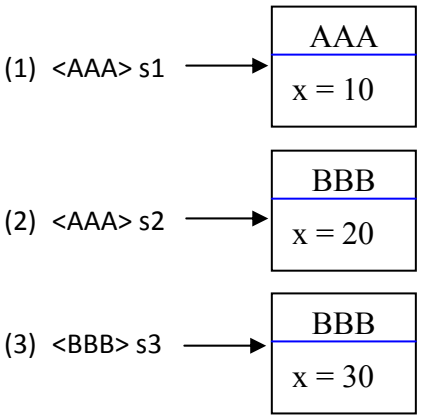
- נשאיר את מחלקה A ללא שינוי
- נשנה את הפעולה הבונה של מחלקה B
  ע"י הוספת זימון של בנאי מחלקת העל עם הערך 6

```
public B (int k)
{
        super(6);
        A a = new A(10);
        System.out.println (k + " ");
}
```

תרגיל 19:

התרגיל בודק ירושה, דריסה, פולימורפיזם של תכונות ופעולות.

א.



(4) מזמן את הפעולה toString של המחלקה AAA ומדפיס H

(5) מזמן את הפעולה toString של המחלקה BBB ומדפיס T

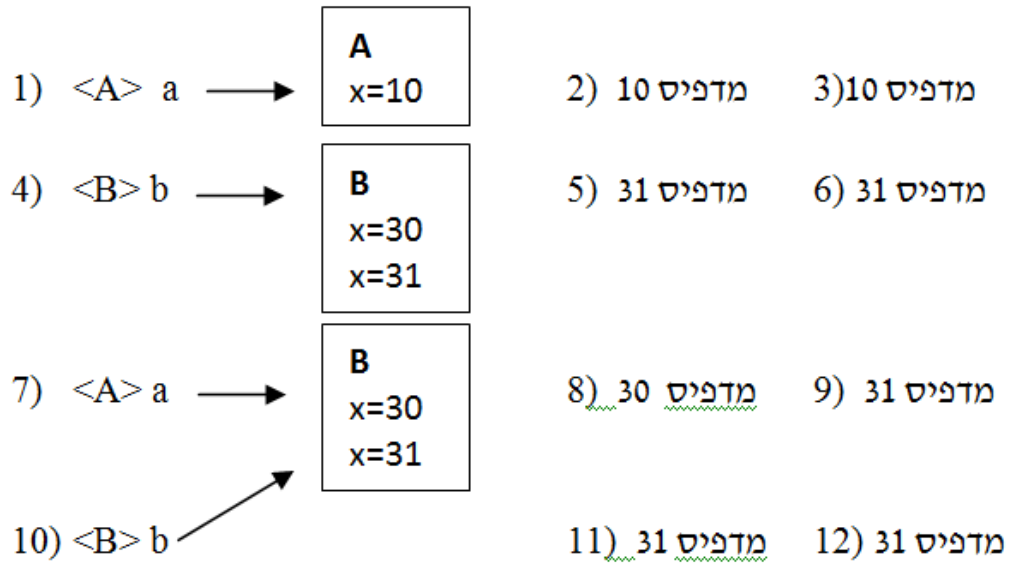
(6) מזמן את הפעולה toString של המחלקה BBB ומדפיס T

(7) למרות שמבצע המרה למחלקת העל, עדיין מזמן את הפעולה toString של המחלקה BBB ומדפיס T

(8) מבצע את הפעולה converToAAA אשר מחזירה עצם מטיפוס המחלקה AAA ולכן מבצע את הפעולה toString של המחלקה AAA ומדפיס H

הפלט הסופי הוא: H T T T H

ב.



<u>הסבר</u>:

(1) ביצירת עצם מטיפוס A מופעלת הפעולה הבונה הריקה אשר מפעילה את הפעולה הבונה עם הפרמטר ולכן הערך של התכונה $x \leftarrow 1*10$

(4) ביצירת עצם מטיפוס B , מופעלת הפעולה הבונה הריקה אשר מזמנת את הפעולה הבונה עם הפרמטר במחלקה B וזו מזמנת את הפעולה הבונה של מחלקת העל עם הערך 2+1=3. הפעולה הבונה של מחלקת העל מאתחלת את הערך x של המחלקה A להיות 3*10=30 ואת הערך x של המחלקה B להיות 30+1=31

(2,3) זימון a.x או a.m() עבור עצם מטיפוס A מחזירות את הערך A של המחלקה ולכן הפלט זהה בשתי השורות.

(5,6) באותו אופן, זימון b.x או b.m() עבור עצם מטיפוס B אשר מסתכלים עליו בנקודת מבט של B יחזיר אותו ערך וזה הערך x של B

(7) כאשר יוצרים עצם מטיפוס B ומציבים אותו במשתנה a מטיפוס מחלקת העל A , הזימון a.x מחזיר את הערך של x במחלקת העל A ואילו a.m() מחזיר את הערך x של המחלקה ממנה נוצר על פי עקרון הפולימורפיזם ולכן יחזיר את הערך של x של המחלקה B .

(10) כאשר ממירים את העצם a , המרה כלפי מטה למחלקה B שוב הזימון b.x יחזיר את הערך x של המחלקה B וכמובן גם b.m() ולכן שוב הערכים זהים.

שאלה 20:

א. (1)

(i) Sport s1 = new Ball();	תקין. Ball סוג של Sport ולכן ניתן ליצור עצם מהטיפוס Ball ולהציב במשתנה מטיפוס מחלקה העל.
(ii) Sport s2 = new Basketball();	תקין. BasketBall הוא סוג של Sport . מבצע המרה כלפי מעלה למחלקת העל.
(iii) Judo b2 = new Sport();	שגוי. Judo הוא לא סוג של Sport ולכן אי אפשר ליצור עצם מטיפוס Sportולהציב במשתנה מטיפוס Judu . חוסר התאמה בטיפוסים.
(iv) Basketball b3 = new Sport();	שגוי. אין התאמה בטיפוסים. Sport הוא לא סוג של BasketBall .

(2) (i) אפשר להגדיר במחלקה **Ball** משתנים מסוג private שאינם מוגדרים במחלקה **Sport** או במחלקה **Judo**. **הטענה נכונה**.

Ball היא תת מחלקה של **Sport** ולכן יש לה את כל התכונות של **Sport** וניתן להוסיף לה תכונות פרטיות אשר לא מוכלות ב **Sport** .

אין קשר בין המחלקות **Ball** ו- Judo ולכן ל- **Ball** יכולות להיות תכונות פרטיות משלה שלא קיימות במחלקה **Judo** .

(ii) אם במחלקה **Sport** יש הגדרה של משתנה בשם title, אמנם המחלקות Ball ו- Judo יורשות אותו אבל לא בהכרח אינן יכולות לגשת אליו ישירות.

גישה לתכונות במחלקת העל תלויה בהרשאות הגישה של התכונות.

במידה ו- title הוגדר כ- private , אכן המחלקות Ball ו- Judo לא יוכלו לגשת אליו ישירות אבל אם יוגדר כ- protected או כ- public , המחלקות Ball ו- Judo יוכלו לגשת אליו ישירות.

ב.

public interface **IHall**
public interface **IGroup**
public class **Sport**
public class **Ball** extends Sport implements IHall
public class **TableTennis** extends Ball
public class **Basketball** extends Ball implements IGroup
public class **Judo** extends Sport implements IHall
public class **RelayRace** extends Sport

ג.

אפשרות 1:

```
public static int count (Sport [] sp)
{
        int counter = 0;
        for (int i = 0 ; i < sp.length ;  i++)
        {
                if (sp[i] instanceof  IHall  && sp[i] instanceof  IGroup )
                        counter++;
        }
        return counter;
}
```

אפשרות 2: מכיוון שהמחלקה Basketball היא היחידה המממשת את שני הממשקים, אפשר גם לכתוב:

```
public static int count (Sport [] sp)
{
        int counter = 0;
        for (int i = 0 ; i < sp.length ; i++)
        {
                if (sp[i] instanceof Basketball)
                        counter ++;
        }
        return counter;
}
```

ב.

פרק ב'

תכנות מונחה עצמים C#

הפתרון לפרק זה נכתב ע"י .....

תרגיל 21:

תרגיל 22:

תרגיל 23:

שאלה 24: