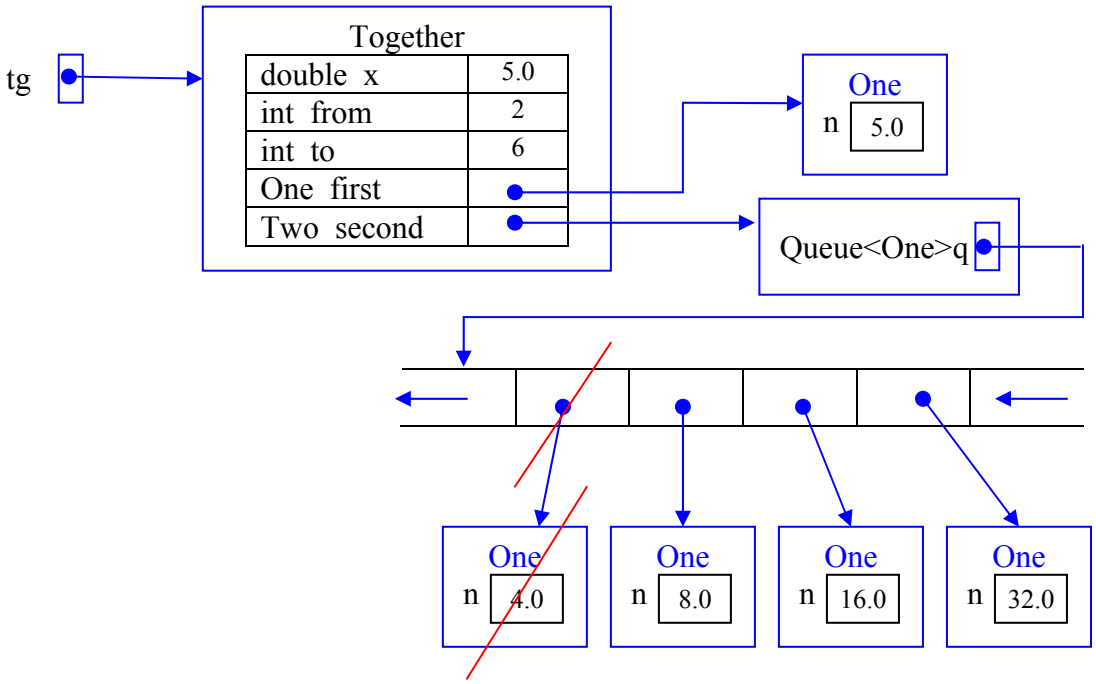
מדעי המחשב ב'

פתרון בחינת הבגרות

פרק א - עיצוב תכנה

שאלה 1:

א.



new Two ((from,to)	n	m	i	i<m	2^i
	2	6	2	T	4
			3	T	8
			4	T	16
			5	T	32
			6	F	

main()	הפעולה	פלט
methodA()	println first.f()	-- MethodA() -- f of One
methodB()	println first,g() second.f()	-- MethodB -- g of One 5.0 f of Two 4.0

ב. אם התור לא ריק, הפעולה מוציאה את האיבר שבראש התור ומדפיסה את ערכו בצירוף הודעה. ואם התור ריק, מדפיסה רק את ההודעה.

אין משמעות להנחה ששני המספרים גדולים מ- 0.
ההנחה צריכה להתייחס למספרים השני והשלישי המקיימים: השלישי גדול מהראשון.

שאלה 2:

a.length = 5

	0	1	2	3	4
a	2	4	7	12	18

א. sod (a,11)

k	i	i<4	j	j<5	a[i]	a[j]	a[i]+a[j] == k	ערך מוחזר
11	0	T	1	T	2	4	F	
			2	T		7	F	
			3	T		12	F	
			4	T		18	F	
			5	F				
	1	T	2	T	4	7	T	אמת

ב. sod (a,10)

k	i	i<4	j	j<5	a[i]	a[j]	a[i]+a[j] == k	ערך מוחזר
10	0	T	1	T	2	4	F	
			2	T		7	F	
			3	T		12	F	
			4	T		18	F	
			5	F				
	1	T	2	T	4	7	F	
			3	T		12	F	
			4	T		18	F	
			5	F				
	2	T	3	T	7	12	F	
			4	T		18	F	
			5	F				
	3	T	4	T	12	18	F	
			5	F				
	4	F						שקר

ג. הפעולה מחזירה "אמת" אם קיימים שני איברים במערך שסכומם k, ו- "שקר" אחרת.

ד. סיבוכיות הפעולה sod היא $O(n^2)$.

בהנחה שיש במערך n איברים:
הלולאה החיצונית רצה על כל המערך - סה"כ n צעדים
ובתוכה לולאה פנימית הרצה בכל פעם על איבר אחד פחות:

$$(n-1)+(n-2)+(n-3)+...+2+1 \Rightarrow S_n = \frac{(1+n-1)(n-1)}{2} \Rightarrow \frac{n^2-n}{2} \Rightarrow O(n^2)$$

ה. what (a, k)

a.length = 5

	0	1	2	3	4
a	2	4	7	12	18

k	left	right	left<right	I a[left]	II a[right]	I+II == k	I+II < k	ערך מוחזר
11	0	4	T	2	18	F	F	
		3	T		12	F	F	
		2	T		7	F	T	
	1		T	4		T		אמת

ו. סיבוכיות הפעולה what היא $O(n)$.

הפעולה עוברת לכל היותר מעבר אחד על כל נתוני המערך, עד שמוצאת או לא מוצאת שני איברים שסכומם k.

ז. סיבוכיות הפעולה sod - ריבועית $O(n^2)$
סיבוכיות הפעולה what - ליניארית $O(n)$
ולכן , what יעילה יותר.

ח. (1) sod תשיג את מטרתה, כי היא בודקת בכל פעם איבר אחר מול כולם.
(2) what לא תשיג את המטרתה, כי היא אינה בודקת את כל האפשרויות.

למשל: עבור המערך הבא תחזיר sod "אמת", ואילו what תחזיר "שקר".

	0	1	2	3	4
a	18	2	4	7	12

שאלה 3:

א.

ההוראה	מצב התור
לאחר האתחול	[]
q.insert (1)	[1]
q.insert (2)	[1, 2]
q.insert (3)	[1, 2, 3]
q.remove()	[2, 3]
q.insert (4)	[2, 3, 4]
q.undo()	[2, 3]
q.undo()	[1, 2, 3]

סעיף א' לאחר הרצת התכנית:

```
start   :   []                  []

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 1
op 1:       [1]                    [(1,1)]

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 2
op 1:       [1, 2]                 [(2,1) , (1,1)]

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 3
op 1:       [1, 2, 3]              [(3,1) , (2,1) , (1,1)]

 0 for remove      1 for insert     2 for undo   -1 to end --> 0
op 0:       [2, 3]                 [(1,0) , (3,1) , (2,1) , (1,1)]

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 4
op 1:       [2, 3, 4]              [(4,1) , (1,0) , (3,1) , (2,1) , (1,1)]

 0 for remove      1 for insert     2 for undo   -1 to end --> 2
op 2:       [2, 3]                 [(1,0) , (3,1) , (2,1) , (1,1)]

 0 for remove      1 for insert     2 for undo   -1 to end --> 2
op 2:       [1, 2, 3]              [(3,1) , (2,1) , (1,1)]

 0 for remove      1 for insert     2 for undo   -1 to end --> -1
```

java:

```
/*
 * Undo  מחלקה המגדירה פעולה עבור
 */
public class Item
{
      private int val;       //  הערך
      private int op;        // remove :0 ,insert :1 הפעולה
}

/*
 *    QueueUndo    תור-ביטול
 */
public class UndoQueue
{
      private Queue <Integer> que;      //  תור הפעולות
      private Stack<Item> stk;          //  מחסנית ה- undo

      public UndoQueue()
      {
            this.que = new Queue<Integer>();
            this.stk = new Stack<Item>();
      }

      //--- הכנסה לתור ביטול ---
      public void insert(int x)
      {
            this.que.insert(x);
            this.stk.push(new Item(x,1));
      }

      //--- הוצאה מתור ביטול ---
      public int remove()
      {
            int x = this.que.remove();
            this.stk.push(new Item(x,0));
            return x;
      }

      //--- undo פעולת ---
      public void undo()
      {
            if(!stk.isEmpty())
            {
                  Item item = this.stk.pop();
                  if (item.getOp() == 0)
                        undoRemove (item.getVal());
                  else
                        undoInsert ();
            }
      }
```

```
//--- הוצאת הערך שנמצא בסוף התור ---
//---          הנחה: התור לא ריק ---
private void undoInsert()
{
      //--- דחיפת סימן לתור ---
      int sign = -1;
      this.que.insert(sign);      //    .כל איברי התור חיוביים
                                  // מספר שלילי מהווה סימן טוב

      //--- העברת כל האיברים פרט לאחרון לסוף התור ---
      int x = this.que.remove();
      boolean finish = false;
      while (! finish)
      {
            if (this.que.head() == sign)
            {
                  this.que.remove();
                  finish = true;
            }
            else
            {
                  this.que.insert(x);
                  x = this.que.remove();
            }
      }
}

//--- הכנסת ערך לתחילת התור ---
private void undoRemove(int x)
{
      Queue<Integer>q = new Queue<Integer>();
      q.insert(x);     // דחיפת הערך לתחילת תור העזר

      //--- העברת כל האיברים לתור העזר ---
      while (! this.que.isEmpty())
            q.insert(this.que.remove());

      //--- החזרת האיברים מתוך העזר לתור המקורי
      while (! q.isEmpty())
            this.que.insert(q.remove());
}
}
```

C#:

```
/*
 * Undo  מחלקה המגדירה פעולה עבור
 */
public class Item
{
      private int val;       //  הערך
      private int op;        // remove :0 ,insert :1 הפעולה
}

/*
 *    QueueUndo    תור-ביטול
 */
public class UndoQueue
{
      private Queue <int> que;    //  תור הפעולות
      private Stack<Item> stk;    //  מחסנית ה- undo

      public UndoQueue()
      {
            this.que = new Queue<int>();
            this.stk = new Stack<Item>();
      }

      //--- הכנסה לתור ביטול ---
      public void Insert(int x)
      {
            this.que.insert(x);
            this.stk.push(new Item(x,1));
      }

      //--- הוצאה מתור ביטול ---
      public int Remove()
      {
            int x = this.que.Remove();
            this.stk.Push(new Item(x,0));
            return x;
      }

      //--- undo פעולת ---
      public void Undo()
      {
            if(!stk.IsEmpty())
            {
                  Item item = this.stk.pop();
                  if (item.GetOp() == 0)
                        UndoRemove (item.GetVal());
                  else
                        UndoInsert ();
            }
      }
```

```
//--- הוצאת הערך שנמצא בסוף התור ---
//---          הנחה: התור לא ריק ---
private void UndoInsert()
{
      //--- דחיפת סימן לתור ---
      int sign = -1;
      this.que.Insert(sign);      //    כל איברי התור חיוביים.
                                  // מספר שלילי מהווה סימן טוב

      //--- העברת כל האיברים פרט לאחרון לסוף התור ---
      int x = this.que.Remove();
      bool finish = false;
      while (! finish)
      {
            if (this.que.Head() == sign)
            {
                  this.que.Remove();
                  finish = true;
            }
            else
            {
                  this.que.Insert(x);
                  x = this.que.Remove();
            }
      }
}

//--- הכנסת ערך לתחילת התור ---
private void UndoRemove(int x)
{
      Queue<int> q = new Queue<int>();
      q.Insert(x);     // דחיפת הערך לתחילת תור העזר

      //--- העברת כל האיברים לתור העזר ---
      while (! this.que.IsEmpty())
            q.insert(this.que.Remove());

      //--- החזרת האיברים מתוך העזר לתור המקורי
      while (! q.IsEmpty())
            this.que.Insert(q.Remove());
}
}
```

פתרון אחר:

מחסנית ה- undo היא מחסנית שכל איבר בה הוא מסוג תור.
בכל פעם שמתבצעת פעולה בתור-ביטול נוצר עותק של התור והוא מוכנס למחסנית.
בכל פעם שמתבצעת פעולת undo מקבל תור-ביטול הפנייה לתור שנשלף מראש המחסנית.

הערה: פתרון זה אינו יעיל כי הוא בזבזני במקום. אחרי n פעולות על התור (ללא undo) יהיה במחסנית n תורים שתופסים הרבה מאוד מקום בזיכרון.

למימוש זה קראתי **QueueUndo** (ולא UndoQueue כנדרש בשאלה)

java:

```
public class QueueUndo
{
      private Queue <Integer> que;            //  תור הפעולות
      private Stack<Queue<Integer>> stk;      //  מחסנית ה- undo

      public QueueUndo()
      {
            this.que = new Queue<Integer>();
            this.stk = new Stack<Queue<Integer>>();     // מחסנית של תורים
      }
      //--- הכנסה לתור ביטול ---
      public void insert(int x) {
            this.stk.push(copyQueue());
            this.que.insert(x);
      }
      //--- הוצאה מתור ביטול ---
      public int remove() {
            this.stk.push(copyQueue());
            int x = this.que.remove();
            return x;
      }
      //--- פעולת undo ---
      public void undo() {
            if(!stk.isEmpty())
                  this.que = this.stk.pop();
      }
      //--- פעולה המחזירה העתק של התור-ביטול ---
      private Queue<Integer> copyQueue() {
            Queue<Integer> q = new Queue<Integer>();
            Queue<Integer> qTmp = new Queue<Integer>();

            while (! this.que.isEmpty())
            {
                  int x = this.que.remove();
                  q.insert(x);
                  qTmp.insert(x);
            }
            while (! qTmp.isEmpty())
                  this.que.insert(qTmp.remove());
            return q;
      }
}
```

C# :

```
public class QueueUndo
{
     private Queue<int> que;                  //  תור הפעולות
     private Stack<Queue<int>> stk;           //  מחסנית ה- undo

     public QueueUndo() {
          this.stk = new Stack<Queue<int>>();      // מחסנית של תורים
          this.que = new Queue<int>();
     }

     //--- הכנסה לתור ביטול ---
     public void Insert(int x) {
          this.stk.Push(CopyQueue());
          this.que.Insert(x);
     }

     //--- הוצאה מתור ביטול ---
     public int Remove()
     {
          this.stk.Push(CopyQueue());
          int x = this.que.Remove();
          return x;
     }

     //--- פעולת undo ---
     public void Undo() {
          if (!stk.IsEmpty())
               this.que = this.stk.Pop();
     }

     //--- פעולה המחזירה העתק של התור-ביטול ---
     private Queue<Integer> CopyQueue() {
          Queue<int> q = new Queue<int>();
          Queue<int> qTmp = new Queue<int>();

          while (!this.que.IsEmpty())
          {
               int x = this.que.Remove();
               q.Insert(x);
               qTmp.Insert(x);
          }
          while (!qTmp.IsEmpty())
               this.que.Insert(qTmp.Remove());
          return q;
     }

}
```

סעיף א' לאחר הרצת התכנית באפשרות - מחסנית של תורים:

```
start   :   []                  []

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 1
op 1:       [1]                 [[]]

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 2
op 1:       [1, 2]                  [[1] , []]

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 3
op 1:       [1, 2, 3]               [[1, 2] , [1] , []]

 0 for remove      1 for insert     2 for undo   -1 to end --> 0
op 0:       [2, 3]                  [[1, 2, 3] , [1, 2] , [1] , []]

 0 for remove      1 for insert     2 for undo   -1 to end --> 1
type a number --> 4
op 1:       [2, 3, 4]               [[2, 3] , [1, 2, 3] , [1, 2] , [1] , []]

 0 for remove      1 for insert     2 for undo   -1 to end --> 2
op 2:       [2, 3]                  [[1, 2, 3] , [1, 2] , [1] , []]

 0 for remove      1 for insert     2 for undo   -1 to end --> 2
op 2:       [1, 2, 3]               [[1, 2] , [1] , []]

 0 for remove      1 for insert     2 for undo   -1 to end --> -1
```

שאלה 4:

java:

```
//---        ט.כניסה: עץ בינארי לא ריק ומחסנית ריקה של מספרים ---
//--- ט.יציאה: המחסנית מכילה את ערכי העלים בסריקה מימין לשמאל ---
public static void leaves (BinTreeNode<Integer>t, Stack<Integer>s)
{
      if (t != null)
      {
            if (isLeaf(t))
                  s.push(t.getInfo());
            else
            {
                  leaves(t.getRight(), s);
                  leaves(t.getLeft(), s);
            }
      }
}

//---        האם עלה?    ---
//-- null אינו bt : הנחה  ---
public static boolean isLeaf (BinTreeNode <Integer> bt)
{
      if (bt.getLeft() == null && bt.getRight() == null)
            return true;
      return false;
}

//---       ט.כניסה: 2 עצים בינאריים המכילים מספרים שלמים ---
//--- ט.יציאה: "אמת" אם מספר וערכי העלים ב-2 העצים שווים ---
//---                                    ו-"שקר" אחרת ---
public static boolean chkTrees ( BinTreeNode <Integer> t1,
                                 BinTreeNode <Integer> t2)
{
      Stack <Integer> s1 = new Stack<Integer>();
      Stack <Integer> s2 = new Stack<Integer>();

      leaves(t1, s1);
      leaves(t2, s2);

      //--- השוואת ערכי העלים שבשתי המחסניות ---
      while (! s1.isEmpty() && ! s2.isEmpty())
            if (s1.pop() != s2.pop())
                  return false;

      //--- אם אחת המחסניות התרוקנה לפני הזמן ---
      if (! s1.isEmpty() || ! s2.isEmpty())
            return false;

      return true;

}
```

C# :

```
//---        ט.כניסה: עץ בינארי לא ריק ומחסנית ריקה של מספרים ---
//--- ט.יציאה: המחסנית מכילה את ערכי העלים בסריקה מימין לשמאל ---
public static void Leaves (BinTreeNode<int>t, Stack<int>s)
{
      if (t != null)
      {
            if (IsLeaf(t))
                  s.Push(t.GetInfo());
            else
            {
                  Leaves(t.GetRight(), s);
                  Leaves(t.GetLeft(), s);
            }
      }
}

//---        ? האם עלה    ---
//-- null אינו bt : הנחה  ---
public static bool IsLeaf (BinTreeNode <int> bt)
{
      if (bt.GetLeft() == null && bt.GetRight() == null)
            return true;
      return false;
}

//---       ט.כניסה: 2 עצים בינאריים המכילים מספרים שלמים ---
//--- ט.יציאה: "אמת" אם מספר וערכי העלים ב-2 העצים שווים ---
//---                                        ו-"שקר" אחרת ---
public static bool ChkTrees (BinTreeNode <int> t1,
                             BinTreeNode <int> t2)
{
      Stack <int> s1 = new Stack<int>();
      Stack <int> s2 = new Stack<int>();

      Leaves(t1, s1);
      Leaves(t2, s2);

      //--- השוואת ערכי העלים שבשתי המחסניות ---
      while (! s1.IsEmpty() && ! s2.IsEmpty())
            if (s1.Pop() != s2.Pop())
                  return false;

      //--- אם אחת המחסניות התרוקנה לפני הזמן ---
      if (! s1.IsEmpty() || ! s2.IsEmpty())
            return false;

      return true;

}
```

פרק ב'

מערכות מחשב ואסמבלר

תרגיל 5:

תרגיל 6:

תרגיל 7:

שאלה 8:

פרק ב'

מבוא לחקר ביצועים

שאלה 9:

שאלה 10:

שאלה 11:

שאלה 12:

פרק ב'

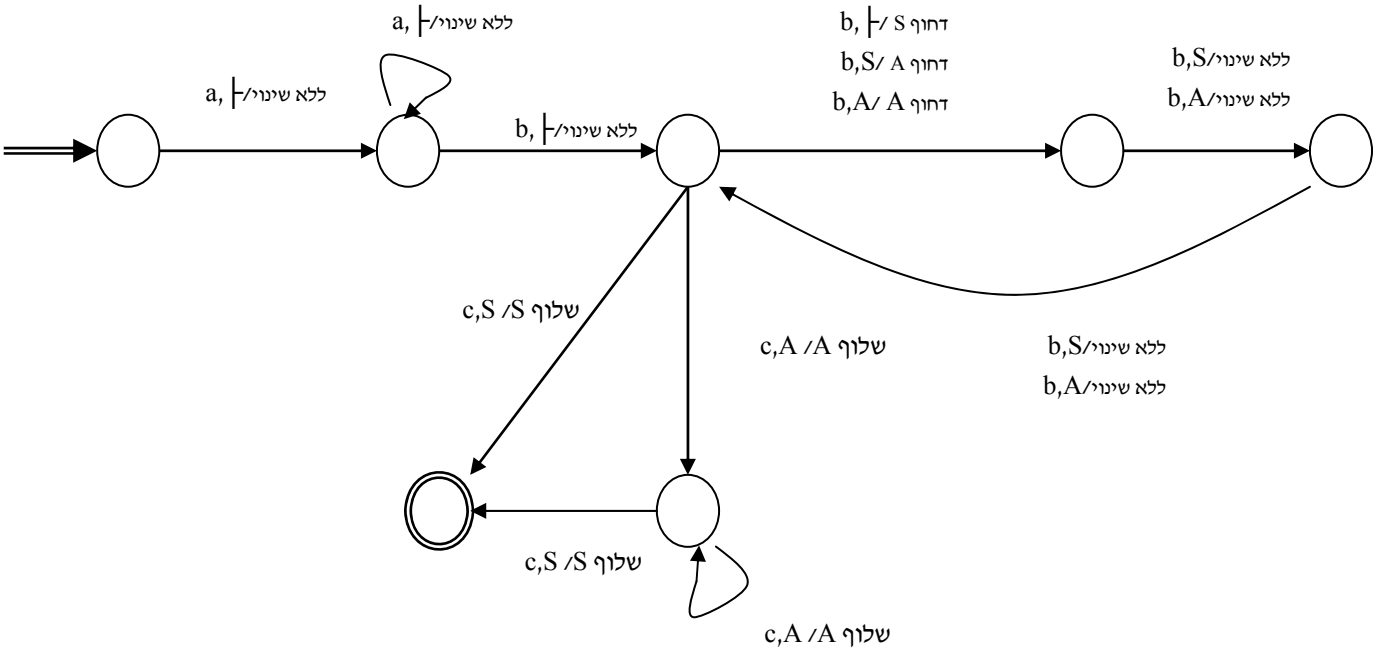
מודלים חישוביים

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

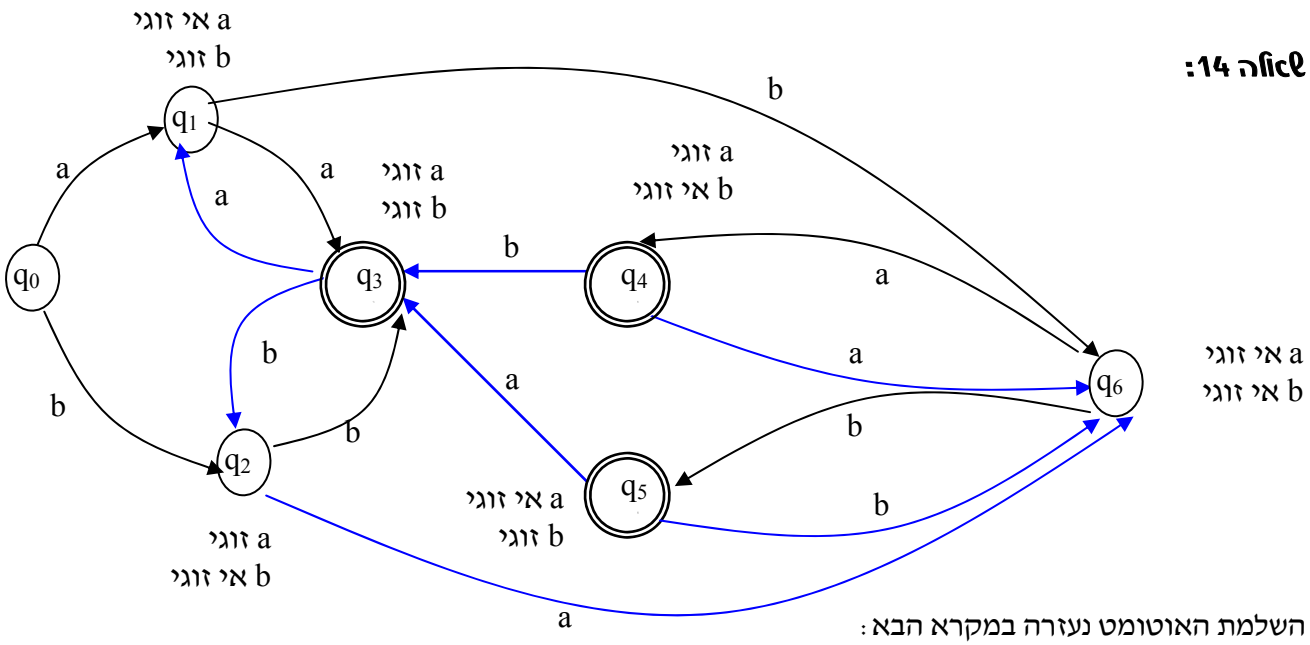
תרגיל 13:

א. המילה הקצרה היא ab^4c עבור n=1 , k=1

ב.



שאלה 14:



השלמת האוטומט נעזרה במקרא הבא:

q_0 - מצב התחלה

q_1 - מספר ה- a אי זוגי ומספר ה- b זוגי.

q_2 - מספר ה- a זוגי ומספר ה- b אי זוגי.

q_3 - מספר ה- a זוגי ומספר ה- b זוגי.

q_4 - מספר ה- a זוגי ומספר ה- b אי זוגי.

q_5 - מספר ה- a אי זוגי ומספר ה- b זוגי.

q_6 - מספר ה- a אי זוגי ומספר ה- b אי זוגי.

ב.

1.

(i) המילה aaba לא מתקבלת.

$$q_0 \xrightarrow{b} q_3 \xrightarrow{b} q_4 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3 \xrightarrow{b} q_4$$

(ii) המילה bbaabb מתקבלת.

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_3 \xrightarrow{a} q_1 \xrightarrow{a} q_2$$

(iii) המילה abaa מתקבלת.

$$q_0 \xrightarrow{b} q_3 \xrightarrow{b} q_4$$

(iv) המילה bb מתקבלת.

2. השפה L המוגדרת ע"י האוטומט היא אוסף כל המילים מעל הא"ב {a,b} שמסתיימות לפחות בשתי אותיות זהות.

שאלה 15:

א. הוכחה שהשפה $L = \{0^n 1^k 2 \mid n > k \geq 0\}$ אינה רגולרית.

נוכיח שהשפה L היא אי רגולרית, בדרך השלילה.
נניח שהשפה רגולרית וקיים אוטומט סופי דטרמיניסטי A הבונה אותה.

תהי הקבוצה האינסופית הבאה: $W = \{0, 0^2, 0^3, ...,0^n,.....\}$

התחלות של מילים בשפה L

נבחר שתי התחלות שונות, $w_1 = 0^i \ , w_2 = 0^j \ \mid i > j, j \geq 0$ מתוך הקבוצה W.

מאחר והאוטומט הוא סופי ניתן להניח כי שתי המילים מגיעות למצב משותף q_t באוטומט A .

משם נשרשר כל מילה עם הרצף $1^j 2$. שתי המילים מגיעות למצב משתף q_r .

המילה $0^i 1^j 2$ עבור i>j שייכת לשפה, ולכן q_r מצב מקבל. מאחר ויש להם מסלול משותף,

יוצא גם ש- $0^j 1^j 2$ גם בשפה, וזה בניגוד לכללי השפה.

לכן הנחתנו ששתי המילים מגיעות למצב משתף אינה נכונה , אלא כל מילה מגיעה למצב אחר. ומאחר והקבוצה W היא אינסופית יוצא שכל מילה מגיע למצב אחר . ומכאן יש אינסוף מצבים ב- A , בניגוד להגדרת אוטומט סופי.

לכן ההנחה שקיים אוטומט סופי הבונה את השפה L אינה נכונה, והשפה אינה רגולרית.

ב. ב. השפה $L_1 \cap L = \phi$.

החיתוך הוא השפה הריקה. השפה L חייבת להתחיל באות 0 (n>0), ואילו השפה L_1 אינה מכילה כלל את הספרה 0.

שאלה 16:

א. מסלולו חישוב עבור f(3)

f(3) = '1'

├	1	1	1	Δ	

	q_0				
├	a	1	1	Δ	

		q_1			
├	a	1	1	Δ	

			q_2		
├	a	1	1	Δ	

				q_2	
├	a	1	1	Δ	

			q_3		
├	a	1	Δ	Δ	

		q_6			
├	a	1	Δ	Δ	

	q_6				
├	1	1	Δ	Δ	Δ

		q_5			
├	1	a	Δ	Δ	Δ

			q_1		
├	1	a	Δ	Δ	Δ

		q_4			
├	1	Δ	Δ	Δ	Δ

q_7

ב. f(5) = '11'

├	1	1	1	1	1	Δ

בסוף הפעולה נקבל

├	1	1	Δ	Δ	Δ	Δ

ג. f(6) = '111'

├	1	1	1	1	1	1	Δ

בסוף הפעולה נקבל

├	1	1	1	Δ	Δ	Δ	Δ

ד. מטרת הפונקציה $f(x) = x/2$ (החלק השלם).
עבור x זוגי נקבל $x/2$ ועבור x אי זוגי נקבל $(x-1)/2$

ה. כדי לקבל את $f(0)$, יש להוסיף את המעבר $q_0 \xrightarrow{\Delta / \Delta, \text{ימין}} q_7$.

אפשרות אחרת: להוסיף את המעבר $q_0 \xrightarrow{\Delta / a, \text{ימין}} q_1$

פרק ב'

תכנות מונחה עצמים Java

תרגיל 17:

תרגיל 18:

תרגיל 19:

שאלה 20:

פרק ב'

תכנות מונחה עצמים C#

הפתרון לפרק זה נכתב ע"י **זיוה קונצמן**.

תרגיל 21:

א. במחלקה AA:

```
public virtual bool IsLike(Object obj
{
        return obj  is AA  &&  ((AA)obj).GetSt().Equals(this.GetSt());
}
```

ב. במחלקה BB:

```
public override bool IsLike(object obj)
{
        return obj is BB  &&  ((BB)obj).GetNum() = = this.GetNum();
}
```

ג. קטע התוכנית נכון. מתבצעת המרה אוטומטית כלפי מעלה של משתנה מטיפוס הבן להפניה מטיפוס האב. הבן הוא גם טיפוס האב לכן אין בעיה.
הפלט יהיה:

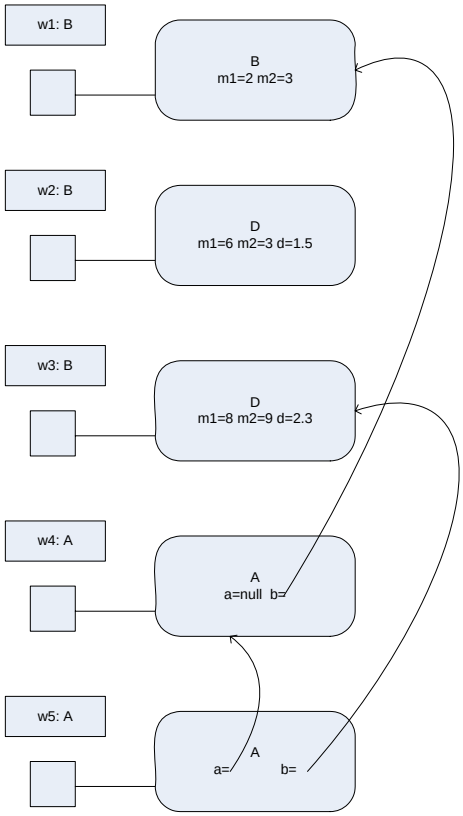
st = excellent num = 1

ד. קטע התוכנית שגוי. לא יכול להמיר כלפי מטה הפניה מסוג האב להיות הפניה מסוג הבן, הוא לא נוצר כ- BB אלא כ- AA, ואינו יכול לעבור המרה למשהו שהוא לא. השגיאה היא שגיאת קומפילציה.

ה.

```
public static string LongString(Object[] a)
{
        string st = "";
        for (int i = 0; i < st.Length; i++)
        {
                if (a[i] is AA && !(a[i] is BB))
                        st += ((AA)a[i]).GetSt();
                else
                        if (a[i] is BB)
                        {
                                for (int j = 1; j < ((BB)a[i]).GetNum(); j++)
                                        st += ((BB)a[i]).GetSt();
                        }
        }
        return st;
}
```

תרגיל 22:

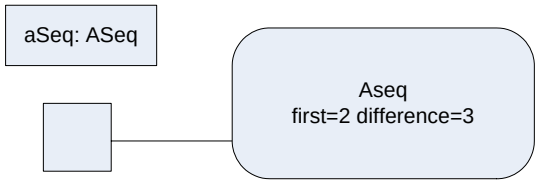


הפלט:

```
B(2,3),#1
B(6,6),#2
D(1.5,6),#1
B(8,9),#3
D(2.3,8,9),#2
A Constructor,#1
A Constructor,#2
```

תרגיל 23:

א.



הפלט:

11

The sequence elements 2, 5, 8, 11, 14

ב. (1)

```
public class Sequence
{
        protected int first;
        public Sequence(int first)
        {
                this.first = first;
        }
        public virtual int TheNElement(int n)
        {
                return this.first;
        }
        public virtual void DisplayNElement(int n)
        {
                Console.Write("The sequence elements");
                for (int i = 0; i < n; i++)
                        Console.Write(this.first + ",");
        }
}
```

(2)

```
public class ASeq:Sequence
{
        private int difference;
        public ASeq(int first, int difference):base(first)
        {
                this.difference = difference;
        }
        public override int TheNElement(int n)
        {
                return this.first + (n - 1) * this.difference;
        }
```

```
        public override void DisplayNElement(int n)
        {
                Console.Write("The sequence elements");
                for (int i = 0; i < n - 1; i++)
                        Console.Write(this.TheNElement(i + 1) + ",");
                Console.WriteLine(this.TheNElement(n));
        }
}
```

ג. אין צורך לעשות כל שינוי. הפעולה תכתב במחלקת Sequence כך:

```
public int SumSeq(int n)
{
        int sum = 0;
        for (int i = 1; i <= n; i++)
                sum += TheNElement(i);
        return sum;
}
```

בשתי המחלקות היורשות אין צורך בכתיבה מחודשת, כל אחת תממש את חישוב האיבר לפי הפעולה הכתובה בה, מכיוון שכאן יופעל מנגנון הדריסה.

ד.

```
public static char Bigger(int n, ASeq a, Gseq g)
{
        int s1 = a.SumSeq(n);
        int s2 = g.SumSeq(n);
        if (s1 > s2)
                return 'A';
        if (s2 > s1)
                return 'G';
        return 'E';
}
```

שאלה 24:

```
public class Vet
{
    private int id;
    private string name;
    private int vetek;

    public void SetVetek()     // עדכון הוותק של הוטרינר ב-1
    {
        this.vetek++;
    }
}


public class Animal
{
    private int numR;
    private string name;
    private string sug;
    private int age;
    private int numVet;
    private Appointement[] lastVisits;

    public void SetAge()                // עדכון הגיל של החיה בשנה
    {
        this.age++;
    }

    public Appointment[] GetLastVisits()
    {
        return this.lastVisits;
    }
}


public class Appointement
{
    private string kodes;
    private int idVet;
}


public class Clinic
{
    private Vet [] veterinarians;
    private Animal [] animals;
```

```
// הפעולה מקבלת סוג חיה ומדפיסה דו"ח הכולל את פרטי כל החיות מסוג זה
public void AllAnimalsSameKind(string kind)
{
    int i = 0;
    while (i < 500 && animals[i] != null)
    {
        if (animals[i].GetType().Equals(kind))
            Console.WriteLine (animals[i].GetName() + "," +
                                animals[i].GetNumR() + "," + animals[i].GetAge());
        i++;
    }
}
// הפעולה מעדכנת בשנה את הותק של כל הוטרינרים
public void UpdateVetek()
{
    int i = 0;
    while (i < 10 && veterinarians[i] != null)
    {
        this.veterinarians[i].SetVetek();
        i++;
    }
}
// הפעולה מעדכנת בשנה את גיל כל החיות
public void UpdateAgeAnimals()
{
    int i = 0;
    while (i < 50 && animals[i] != null)
    {
        this.animals[i].SetAge();
        i++;
    }
}
// הפעולה מקבלת ת.ז. של וטרינר ומחזירה את שמו
public string GetVet(int id)
{
    int i = 0;
    while (i < 10 && veterinarians[i] != null)
    {
        if (veterinarians[i].GetId() == id)
            return veterinarians[i].GetName();
        i++;
    }
    return null;
}
```

```
        // הפעולה מקבלת חיה, וטרינר מטפל וקוד טיפולים נוכחי ומוסיפה את
        // הביקור למאגר הביקורים הקיים של חיה זו
        public void AddAppointment(Animal p, string t, Vet v)
        {
                int i = 0;
                while (i < 50 && !animals[i].Equals(p))
                        i++;
                Appointment[] a = animals[i].GetLastVisits();
                i = 0;
                while (a[i] != null)
                        i++;
                a[i] = new Appointment(v, t);
        }
}
```