מבחן 2009

מבחן בגרות 2009

פרטים כלליים

מועד הבחינה: בכל זמן

מספר השאלון: 1

משך הבחינה: 3 שעות

חומר עזר בשימוש: הכל (ספרים ומחברות)

המלצות: קרא המלצות לפני הבחינה ובדיקות אחרונות לפני מסירה (עמודים 8-11)

מבנה השאלון

פרק ראשון - עיצוב תוכנה	2 שאלות (בחירה מ-4) לכל שאלה – 25 נקודות	סה"כ 50 נקודות
פרק שני - מודלים חישוביים	2 שאלות (בחירה מ - 4) לכל שאלה – 25 נקודות	סה"כ 50 נקודות

תוכן עניינים של פתרון המבחן

מבחן 2009 פתרון מבחן בגרות, קיץ 2009

פתרון פרק ראשון - ג'אווה תוכנה 2009

פתרון שאלה 1

נושא מרכזי: רשימות
סוג השאלה: פיתוח פעולה

נתוני השאלה:

רשימה l תיקרא משולשת אם היא מקיימת את התנאים האלה:

- הרשימה אינה ריקה.
- מספר האיברים בה מתחלקים ב-3 ללא שארית.
- האיברים בשליש הראשון של הרשימה מכילים את אותם ערכים שמכילים האיברים בשליש השני של הרשימה ואותם ערכים שמכילים האיברים בשליש השלישי של הרשימה. הערכים מסודרים באותו סדר בכל אחד מהשלישים.

הדרישה: לכתוב פעולה המקבלת רשימה שהאיברים שלה מטיפוס שלם.
אם הרשימה משולשת, הפעולה מחזירה true. אחרת- הפעולה מחזירה false.

מספר פתרונות

גרסת פתרון 1

הסבר פתרון:

נעזר בשתי פעולות: פעולה המחזירה אורך של רשימה, ופעולה המקבלת רשימה ומיקום של איבר ברשימה, ומחזירה הפנייה לאותו איבר.
נמנה את מספר האיברים ברשימה. אם מספר האיברים אינו מתחלק ב-3 הרשימה אינה משולשת. אחרת, נמקם 3 הפניות לתחילת כל שליש ברשימה. נרוץ במקביל על שלושת השלישים ונבדוק שוויון בין האיברים. אם התגלה אי-שוויון במהלך הסריקה, הפעולה תחזיר false.

```
import java.util.*;
   class  Q1A
   {
       static Scanner reader = new Scanner (System.in);
      public static boolean isTriple(List<Integer> l)
      { //  הפעולה מקבלת רשימה ומחזירה true אם הרשימה משולשת או false אחרת
         int len = getLength(l);
         if (len == 0 || len % 3 != 0)
            return false;
         Node<Integer> pos1 = l.getFirst();                  // מיקום תחילת השליש הראשון
         Node<Integer> pos2 = getPosition(l, len / 3);       // מיקום תחילת השליש השני
         Node<Integer> pos3 = getPosition(l, 2 * len / 3); // מיקום תחילת השליש השלישי
         while (pos3 != null)
         {
            if (pos1.getInfo() != pos2.getInfo() || pos2.getInfo() != pos3.getInfo())
               return false;
            pos1 = pos1.getNext();
            pos2 = pos2.getNext();
            pos3 = pos3.getNext();
```

```
    }
    return true;
}
private static int getLength(List<Integer> l)
{
  // הפעולה מקבלת רשימה ומחזירה את אורכה
    Node<Integer> pos = l.getFirst();
    int len = 0;
    while (pos != null)
    {
       len++;
       pos = pos.getNext();
    }
    return len;
}
..... .....
private static Node<Integer> getPosition(List<Integer> l, int place)
{
    // הפעולה מקבלת רשימה ומיקום איבר ברשימה
    // הפעולה מחזירה הפנייה לחוליה שזהו מיקומה
    Node<Integer> pos = l.getFirst();
    for (int i = 0; i < place ; i++)
    {
       pos = pos.getNext();
    }
    return pos;
}
```

גרסת פתרון 2

<u>הסבר פתרון:</u>

נעזר בשתי פעולות: פעולה המחזירה אורך של רשימה, ופעולה המקבלת רשימה ומיקום של איבר ברשימה, ומחזירה הפנייה לאותו איבר.

נמנה את מספר האיברים ברשימה. אם מספר האיברים אינו מתחלק ב-3 הרשימה אינה משולשת. אחרת, נמקם שתי הפניות: לתחילת השליש הראשון של הרשימה ולתחילת השליש השני של הרשימה. נרוץ במקביל עם שתי ההפניות עד שההפניה של השליש השני תגיע לסוף הרשימה. בדרך זו יבדקו שלושת השלישים מכיוון שההפניות ירוצו במקביל על שליש ראשון ושני, ולאחר מכן במקביל על שליש שני ושלישי. אם התגלה אי-שוויון במהלך הסריקה, הפעולה תחזיר false.

```
import java.util.*;
  class  Q1B
  {
    static Scanner reader = new Scanner (System.in);
    public static boolean isTriple (List<Integer> l)
    { //  הפעולה מקבלת רשימה ומחזירהtrue אם הרשימה משולשת או false אחרת
      int len = getLength(l);
      if (len == 0 || len % 3 != 0)
        return false;
      Node<Integer> pos1 = l.getFirst();
```



```
        Node<Integer> pos2 = getPosition(l, len / 3);

        while (pos2 != null)
        {
            if (pos1.getInfo() != pos2.getInfo() )
                return false;
            pos1 = pos1.getNext();
            pos2 = pos2.getNext();
        }
        return true;
    }

    private static int getLength (List<Integer> l)
    {
        Node<Integer> pos = l.getFirst();
        int len = 0;
        while (pos != null)
        {
            len++;
            pos = pos.getNext();
        }
        return len;
    }

    private static Node<Integer> getPosition (List<Integer> l, int place)
    {
        Node<Integer> pos = l.getFirst();
        for (int i = 0; i < place; i++)
        {
            pos = pos.getNext();
        }
        return pos;
    }
```

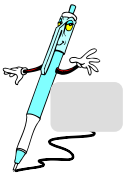
פתרון שאלה 2

נושא מרכזי: עצים
סוג השאלה: פיתוח פעולה, בניית עץ לפי סריקות

א. יש לכתוב פעולה בוליאנית המקבלת שורש של עץ בינארי ובודקת האם הוא עץ "ימין-שמאלי".
כלומר, אם לכל צומת בעץ שיש לו בן ימני יש לו גם בן-שמאלי.

```
public static boolean rightLeft (BinTreeNode<Integer> t)
{
    if (t == null || (t.getLeft() == null t.getRight ==null) )  // עץ עלה – קיים מסלול אחיד
        return true;
    if (t.getRight() != null)          // אם יש בן ימני
        if (t.getLeft() == null)  // אזי אם אין בן שמאלי
            return false;             // אזי המצב לא תקין
    return (rightLeft(t.getRight()  && (rightLeft(t.getLeftt()    // המשך בדיקה על כל העץ
```

}



הסבר הפתרון: עבור כל צומת בעץ, יש לבדוק אם יש לו בן ימני אזי חייב להיות לו גם בן שמאלי. אם זה מתקיים יש להמשיך ולבדוק את קיום התנאי על תת העץ השמאלי וגם על תת העץ הימני.

מבנה הפעולה:

- אם הצומת הנוכחי לא קיים או שהוא עלה, אזי הסריקה הסתיימה באופן תקין.
- בדיקה האם יש לצומת בן ימני וגם בן שמאלי, אם יש ימני ואין שמאלי מוחזר false.
- הבדיקה הכללית תצליח אם התנאי מתקיים על הצומת הנוכחי (נבדק בשלב הקודם) וגם על תת העץ השמאלי ועל תת העץ הימני.

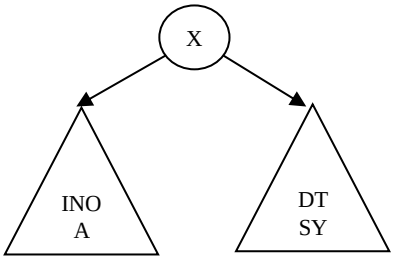
ב. בשאלה נתונים סדר סריקה תחילי של עץ וסדר סריקה תוכי של אותו עץ. נדרש לתאר את העץ ולהציג את סדר הסריקה הסופי שלו.

הסריקה התחילית הנתונה: XAIONYTDS

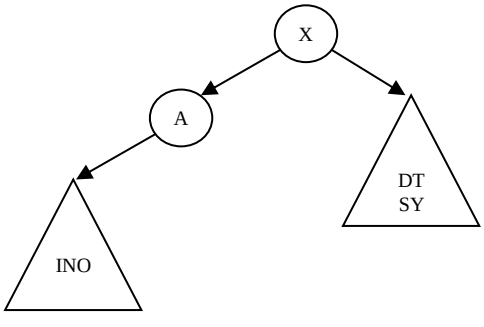
הסריקה התוכית הנתונה: INOAXDTSY

בהינתן סריקה תוכית וסריקה תחילית של עץ, קיים רק עץ אחד שיכול להתאים לתיאור. דרך הבדיקה היא עקבית ומבוצעת באופן זה:

- הצומת הראשון שנסרק בסריקה תחילית הוא שורש העץ – לכן השורש הוא: X
- בסריקה תוכית, כל תת-עץ שמאל נסרק לפני השורש, וכל תת-עץ ימין נסרק אחריו, לכן נסתכל על הסריקה התוכית, נמצא בתוכה את השורש X ובהתאמה נפריד ונראה ש: INOA הם מצד שמאל של X, ו DTSY הם מצד ימין של X.

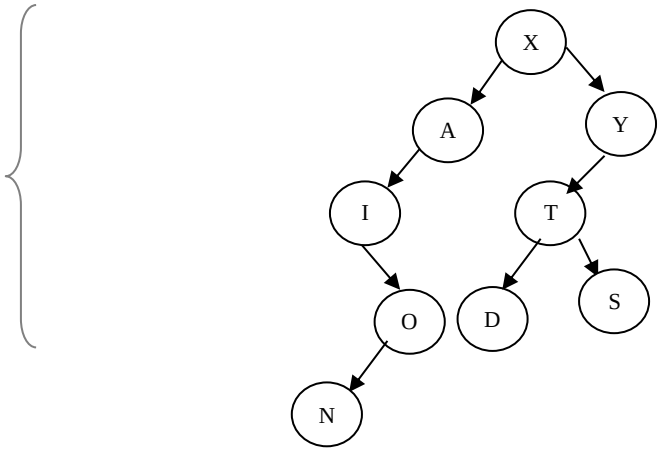


- באותו אופן נסתכל על הצמתים שזיהינו בצד שמאל של X. לפי הסריקה התחילית נזהה שהשורש הוא A, לפי הסריקה התוכית נראה שיתר הצמתים הם בצד שמאל של A, כי מימינו לא מופיע אף צומת. לפיכך המצב כעת הוא:



- בתוך הרצף ION, לפי סריקה תחילית, I הוא השורש, לכן I שמאלי ל- A. לפי הסריקה התוכית, NO מימין ל- I. שוב נסתכל על התחילית ונראה ש- O הוא השורש, ולפי התוכית N

משמאלו. אותו היסק נבצע גם על תת העץ הימני ומתקבל העץ: סריקה סופית של העץ היא:
NOIADSTYX



פתרון שאלה 3

נושא מרכזי: תור
סוג השאלה: ניתוח פעולה, מעקב

נתוני השאלה:

נתונות שתי פעולות:
פעולה ראשונה: מקבלת תור לא ריק המכיל מספרים שלמים.
פעולה שנייה: מקבלת מספר גדול מ-0 או שווה לו.
נתון התור myQueue מטיפוס Queue<int>

תשובות לסעיפים א-ה:

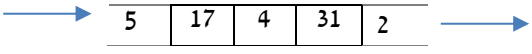
א. הזימון sod1(myQueue) יחזיר את המספר 2, תוך כדי הפיכת סדר האיברים בתור.

מס זימון	i	result	j	q	return
1)	5	5	sod1(q)	2 | 31 | 4 | 17 | 5	
2)	17	17	sod1(q)	2 | 31 | 4 | 17	
3)	4	4	sod1(q)	2 | 31 | 4	
4)	31	31	sod1(q)	2 | 31	
5)	2	2		2	2

4)	31	31 2	2	31 | 2	2
3)	4	4 2	2	4 | 31 | 2	2
2)	17	17 2	2	17 | 4 | 31 | 2 |	2
1)	5	5 2	2	5 | 17 | 4 | 31 | 2	2

חזרה מזימונים רקורסיביים

התור בתום המעקב:



ב. הפעולה sod1(queue) בעבור תור queue לא ריק מטיפוס Queue<int> מחזירה את המספר הקטן ביותר בתור והופכת את אברי התור.

ג. מעקב עבור הזימון sod2(17852):

i	i == 0	a	b	a > b	return
17852	false	2	sod2(1785) 8	false	8
1785	false	5	sod2(178) 8	false	8
178	false	8	sod2(17) 7	true	8
17	false	7	sod(1) 1	true	7
1	false	1	sod(0) 0	true	1
0	true				0

הזימון sod2(17852) יחזיר את הספרה 8.

ד. הפעולה sod2(k) בעבור מספר k גדול מ-0 מטיפוס שלם מחזירה את הספרה הגדולה ביותר במספר.

ה. הפעולה מחזירה את הספרה הגדולה ביותר במספר הקטן ביותר בתור והופכת את סדר אברי התור.

פתרון שאלה 4

נושא מרכזי: טיפוסים, רשימות
סוג השאלה: פיתוח מחלקה ופיתוח פעולות

נתוני השאלה:

תיאור של יומן אלקטרוני לניהול פגישות. היומן מכיל את הימים של שנה אחת.
כל יום מיוצג על-ידי :

- תאריך, הכולל חודש, יום בחודש.
- רשימת הפגישות באותו יום. ...

נתון תיאור חלקי של המחלקה פגישה – Meeting (ראה שאלון 2010).
נתון חלק ממשק המחלקה יום ביומן.

הדרישה:

א. לכתוב את כותרת המחלקה DayInSchedule, ואת התכונות שלה.
ב. לממש את הפעולה canStart המוצגת בממשק של המחלקה DayInSchedule.
ג. לממש פעולה חיצונית שתקבל רשימה של ימים ביומן, ופגישה. הפעולה תדפיס את החודש והיום בחודש של כל אחד מהימים ברשימה, שבהם אפשר לשבץ את הפגישה.....

```
public class  DayInSchedule
{
    private int day;            // יום
    private int month; //חודש
    private List<Meeting> listMetting; //רשימת פגישות

    public List<Integer> getFreeHours()
    {
       return new List<Integer>();
    }

    public int getDay()
    {
        return this.day;
    }

     public int  getMonth()
    {
        return this.month;
    }
    public boolean canStart(int startHour, int minutes)
    { // הפעולה מקבלת שעת התחלה ומשך זמן הפגישה
      // הפעולה מחזירה אמת אם אפשר להתחיל את הפגישה ושקר אחרת
      List<Integer> freeHours = this.getFreeHours(); // רשימת כל השעות הפנויות
      Node<Integer> pos = freeHours.getFirst();
      int  time = minutes;
      if (time % 60 != 0)          // מספר לא שלם של שעות
          time = (time / 60) + 1;
      else
          time = time / 60;
      time = time + startHour;
```

```
        while (pos != null && pos.getInfo() < startHour)
           pos = pos.getNext();    // התקדמות עד לשעת ההתחלה או עד סוף הרשימה
        if (pos == null || pos.getInfo() > startHour)
           return false;                // לא קיימת שעה פנויה
        int nexthour = startHour + 1;
        pos = pos.getNext();
        while (pos != null && nexthour < time)
        {
           if (pos.getInfo() != nexthour)
              return false;
           nexthour++;
           pos = pos.getNext();
        }
        return true;
     }
  }
```

הסבר פתרון לפעולה canStart :

נחשב את משך הפגישה בשעות. אם משך הפגישה אינו שעות שלימות יש צורך להוסיף שעה עבור הדקות החורגות מאחר ופגישות מתקיימות רק בשעות שלמות. נחשב את שעת הסיום על-ידי סכום של שעת ההתחלה ומשך הפגישה. נסרוק את הרשימה כדי לבדוק אם שעת ההתחלה פנויה. אם לא – לא ניתן לקיים את הפגישה. אחרת, נסרוק את המשך הרשימה כדי לבדוק אם כל השעות שבהן צריכה להתקיים הפגישה פנויות ונחזיר אמת/שקר בהתאם.

```
class  MainQue4
{
     public static void PrintAvailbleDay (List<DayInSchedule> lst, Meeting m)
     {
        Node<DayInSchedule> p = lst.getFirst();
        while (p != null)
        { // מעבר על כל הימים ביומן
           if (p.getInfo().canStart (m.getStartHour(), m.getMinutes()))
           { // אם אפשר לשבץ את הפגישה
                 System.out.println (p.getInfo().getDay());
                 System.out.println (p.getInfo().getMonth());
           }
           p = p.getNext();
        }
     }
 }
```


פתרון פרק שני - מודלים חישוביים 2009

פתרון שאלה 13

נושאים מרכזיים: רגולריות - אוטומט לא דטרמיניסטי
סוג השאלה: ניתוח אוטומט, הוכחת רגולריות

א. נתון אוטומט דטרמיניסטי לא מלא A (ראה שרטוט בשאלון).

i. המילים הבאות מתקבלות/לא מתקבלות על-ידי האוטומט A:

(1) המילה babbaaa בשפה. להלן המסלול המקבל:

$$q_0 \xrightarrow{b} q_9 \xrightarrow{a} q_1 \xrightarrow{b} q_9 \xrightarrow{b} q_8 \xrightarrow{a} q_7 \xrightarrow{a} q_6 \xrightarrow{a} q_5$$

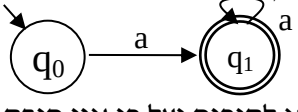
(2) המילה aababaaa לא בשפה.

(3) המילה aaaabbba בשפה. להלן המסלול המקבל:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3 \xrightarrow{b} q_4 \xrightarrow{b} q_5 \xrightarrow{b} q_5 \xrightarrow{a} q_6$$

ii. המילים הקצרות ביותר המתקבלות על-ידי האוטומט A הן: aaabb, bbaaa.

ב. השפה $L_1 \cap L_2$ היא שפה רגולרית. $L_1 \cap L_2 = \{a^n \mid n \geq 1\}$
השפה מכילה מילים הבנויות מרצף של a-ים.
שפה זו ניתנת לתיאור על-ידי אוטומט סופי דטרמיניסטי מלא (או לא מלא) מאחר ונדרש מצב ה"זוכר" מופע אחד לפחות של a וממנו מתקבלות רק אותיות a.

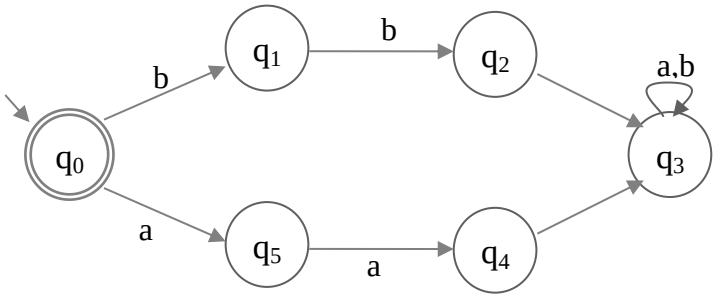


שים ♥: בשפה זו התבקשת לנמק אי-רגולריות ולא להוכיח ועל כן אין חובה לבנות אוטומט.

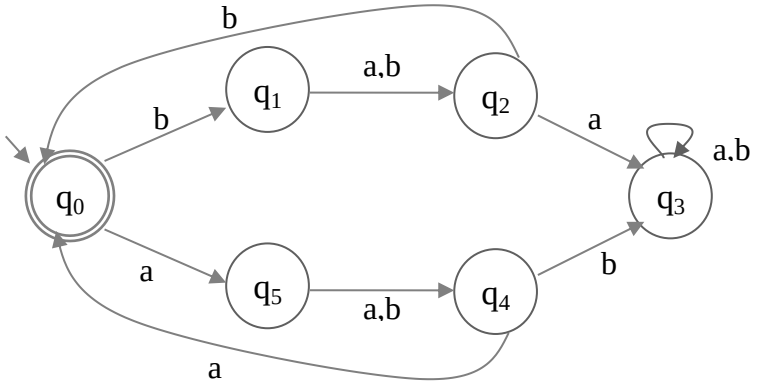
פתרון שאלה 14

נושא מרכזי: אוטומט סופי דטרמיניסטי
סוג השאלה: השלמת אוטומט

נתוני השאלה:
מילה באורך 3 תיקרא פלינדרום באורך 3, אם התו הראשון במילה שווה לתו האחרון במילה.
נתונה השפה L הבאה: $L = \{w \mid$ w היא שרשור של 0 או יותר פלינדרומים באורך 3$\}$
נתון ציור חלקי של אוטומט סופי דטרמיניסטי המקבל את השפה L.



א. האוטומט המלא שמקבל את השפה L:



ב. השפה היא: $L\cap\{(aab)^n \mid n\geq 0\} = \{\varepsilon\}$

נימוק: השפה $(aab)^n$ בנויה משלשלות של הרצף aab שאינו פלינדרום. לכן, המילה המשותפת היחידה לשתי השפות היא המילה הריקה.

פתרון שאלה 15

נושא מרכזי: שפות רגולריות
סוג השאלה: הוכחת אי-רגולריות

א. השפה $L_1\cap L_2$ היא: $L_1\cap L_2 = 0^n1^n \mid n\geq 1$

ב. השפה $Init(L_3)$ היא:

$Init(L_3) = \{u \mid uv\in L_3,\ u,v\in\Sigma^*\} =$
$\{0^i1^{i-j}\cdot 1^j \mid i\geq 0,\ 0\leq\leq i\}$
$u = \{0^i1^{i-j} \mid 0\geq j\leq 1\}$

הסבר: $Init(L_3)$ אלה למעשה תת- המילים שהן התחלות (רישיות) במילים השייכות לשפה L_3. מאחר וכל מילה בשפה היא רצף 0 –ים שאחריו רצף 1-ים כאשר מספר המופעים של 0 שווה למספר מופעי ה-1, ניתן לחלק כל מילה להתחלה (רישא) וסיומת (סיפא).

נגדיר את ההתחלות כרצף של 0-ים, שאחריו רצף 1-ים הקטן ממנו כך שהסיפא תהיה רצף של 1-ים המשלים את מספר המופעים של 1 ברישא כך שיהיה שווה למספר ה-0-ים. לכן,

$u = 0^i1^{i-j} \mid 0\geq j\leq 1$
$v = 1^j \mid j\geq 0$

ג. השפה $Fin(L_3)$ היא: $Fin(L_3) = 1^j$

הסבר – ראה סעיף ב לעיל.

ד. $0011\notin Min(L_4)$

נימוק: כל מילה השייכת לשפה $Min(L_4)$ צריכה להיות שרשור של 2 תתי מילים w_1,w_2 כך ש- $w_1\notin L_4$ ו- $w_2\neq\varepsilon$. המילה 0011 יוצרת את השרשורים הבאים כך ש- $w_2\notin\varepsilon$:

{ ε·0011, 0·011, 00·11, 001 ·1 }

כדי שהמילה תהיה שייכת לשפה, כל השרשורים צריכים לקיים את התנאי. השרשור ε·0011 אינו מקיים את התנאי $w_1 \notin L_4$ מאחר ו$\varepsilon \in$ -L_4

ה. $0011 \notin Min(L_5)$

נימוק: (בדומה לסעיף ד לעיל)

כל מילה השייכת לשפה $Min(L_5)$ צריכה להיות שרשור של 2 תתי מילים w_1,w_2 כך ש- $w_1 \notin L_5$ ו- $w2 \notin \varepsilon$.

השרשורים היוצרים את המילה 0011 כאשר $w_2 \notin \varepsilon$ הם: { ε·0011, 0·011, 00·11, 001 ·1 }

כל אחת מההתחלות (רישות) שייכת לשפה L_5 ועל-כן המילה 0011 אינה שייכת ל- $Min(L_5)$.

ו. השפה$L_4 \cap L_5$אינה רגולרית.

$L_4 \cap L_5 = \{ 0^i1^k \mid 0 \le i \le k \} \cap \{ 0^i1^k \mid 0 \le k \le i \} = \{ 0^i1^i \mid i \ge 0 \}$

נימוק: מספר ה-0-ים בתחילת המילה צריך להיות שווה למספר ה- 1-ים בסוף המילה. מאחר ויש תלות של מנייה בין מספר מופעי ה- 0 למספר מופעי ה-1 ויש אינסוף אפשרויות, לא ניתן לבנות לשפה מודל המתאר שפה רגולרית.

פתרון שאלה 16

נושאים מרכזיים: מכונת טיורינג
סוג השאלה: ניתוח מכונה

נתונה מכונת טיורינג...

א. לאחר חישוב f(5) הסרט יכיל:

├	1	1	1	1	1	1	$	△	...

ב. לאחר חישוב f(6) הסרט יכיל:

├	1	1	1	1	1	1	$	△	...

ג. הפונקציה f(x) שהמכונה מחשבת היא:

f(x) = x+x%2

כלומר, אם מספר ה-1-ים אי-זוגי יתווסף 1 ואם מספר ה-1-ים זוגי יישאר כפי שהיה.