

# 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

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

## תוכן עניינים של פתרון המבחן

### פרק ראשון - עיצוב תוכנה

שאלה 1: רשימות [ פיתוח פעולה ] .....

שאלה 2: עצים [ פיתוח פעולה, בניית עץ לזי סריקות ] .....

שאלה 3: תור [ ניתוח פעולה – מעקב וכו' ] .....

שאלה 4: טיפוסים, רשימות [ פיתוח מחלקה ופיתוח פעולות ] .....

### פרק שני - מודלים חישוביים

שאלה 13: רגולריות- אוטומט לא דטרמיניסטי [ניתוח אוטומט, הוכחת רגולריות] ...

שאלה 14: שפות רגולריות, פעולות על מילים ושפות [ השלמת מודל ] .....

שאלה 15: שפות רגולריות, פעולות על מילים ושפות [ נימוק נכונות של טענות ] .....

שאלה 16: מכונת טיורינג [ניתוח מכונה] .....

### 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.



```
using System;
class Q1A
{
```

```
    public static bool IsTriple(List<int> 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<int> pos1 = l.First(); // מיקום תחילת השליש הראשון
        Node<int> pos2 = GetPosition(l, len / 3); // מיקום תחילת השליש השני
        Node<int> 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<int> l)
{
    // הפעולה מקבלת רשימה ומחזירה את אורכה
    Node<int> pos = l.GetFirst();
    int len = 0;
    while (pos != null)
    {
        len++;
        pos = pos.GetNext();
    }
    return len;
}

```

```

private static Node<int> GetPosition(List<int> l, int place)
{
    // הפעולה מקבלת רשימה ומיקום איבר ברשימה
    // הפעולה מחזירה הפנייה לחוליה שזהו מיקומה
    Node<int> pos = l.GetFirst();
    for (int i = 0; i < place ; i++)
    {
        pos = pos.GetNext();
    }
    return pos;
}

```

## גרסת פתרון 2

### 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 שתי הפניות: לתחילת השליש הראשון של הרשימה ולתחילת השליש השני של הרשימה. נרוץ במקביל עם שתי הפניות עד שהפניה של השליש השני תגיע לסוף הרשימה. בדרך זו יבדקו שלושת השלישים מכיוון שההפניות ירוצו במקביל על שלישי ראשון ושני, ולאחר מכן במקביל על שלישי שני ושלישי. אם התגלה אי-שוויון במהלך הסריקה, הפעולה תחזיר false.



```

using System;
class Q1B
{

    public static bool IsTriple (List<int> 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<int> pos1 = l.GetFirst();
    }
}

```

```

Node<int> pos2 = GetPosition(1, 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<int> l)
{
    Node<int> pos = l.GetFirst();
    int len = 0;
    while (pos != null)
    {
        len++;
        pos = pos.GetNext();
    }
    return len;
}

private static Node<int> GetPosition (List<int> l, int place)
{
    Node<int> 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.

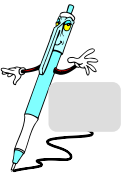
```

public static bool RightLeft (BinTreeNode<int> t)
{
    if (t == null || (t.GetLeft() == null t.GetRight == null) ) // 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.

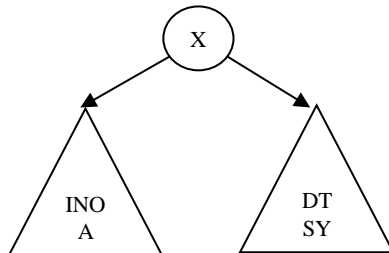
ב. בשאלה נתונים סדר סריקה תחילי של עץ וסדר סריקה תוכי של אותו עץ. נדרש לתאר את העץ ולהציג את סדר הסריקה הסופי שלו.

הסריקה התחילית הנתונה: XAIONYTDS

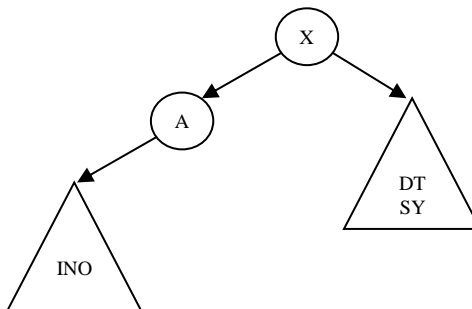
הסריקה התוכית הנתונה: INOAXDTSY

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.



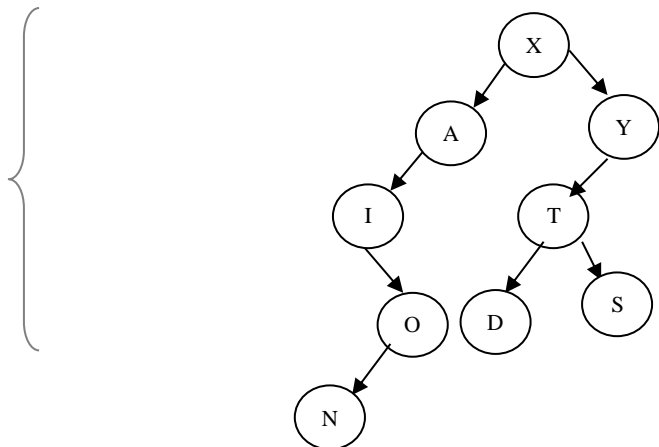
- באותו אופן נסתכל על הצמתים שזיהינו בצד שמאל של X. לפי הסריקה התחילית נזהה שהשורש הוא A, לפי הסריקה התוכית נראה שיתר הצמתים הם בצד שמאל של A, כי מימינו לא מופיע אף צומת. לפיכך המצב כעת הוא:



- 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.

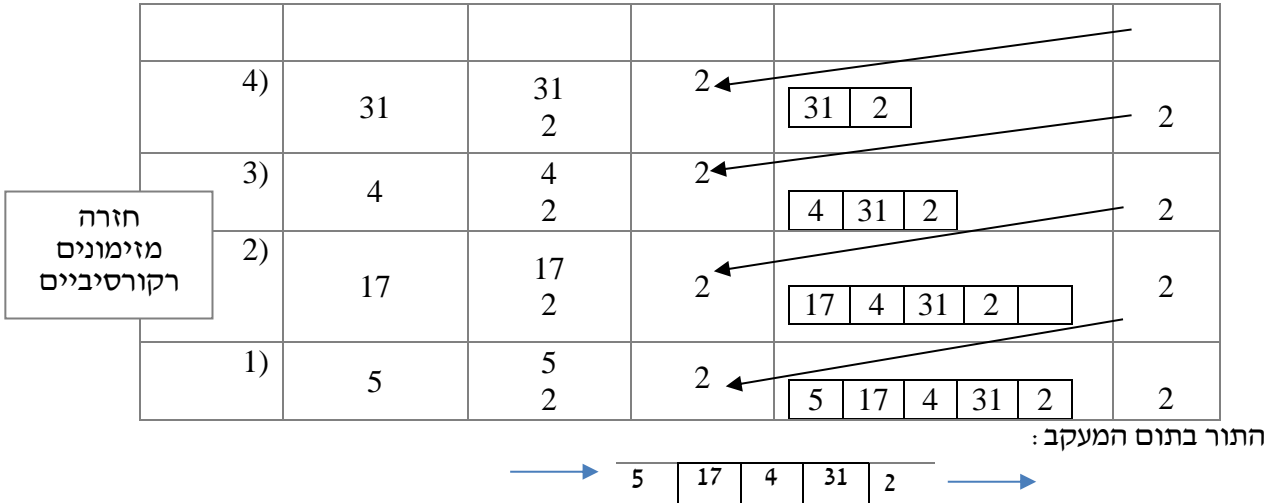
פעולה שנייה: מקבלת מספר גדול מ-0 או שווה לו.

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



חזרה  
מזימונים  
רקורסיביים

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. היוםן 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 מממשק המחלקה יום ביומן.

### הדרישה:

- לכתוב את כותרת המחלקה DayInSchedule, ואת התכונות שלה.
- לממש את הפעולה CanStart המוצגת בממשק של המחלקה DayInSchedule.
- 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<int> GetFreeHours()
    {
        return new List<int>();
    }

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

    public int GetMonth()
    {
        return this.month;
    }

    public bool CanStart(int startHour, int minutes)
    { // הפעולה מקבלת שעת התחלה ומשך זמן הפגישה
        // הפעולה מחזירה אמת אם אפשר להתחיל את הפגישה ושקר אחרת
        List<int> freeHours = this.GetFreeHours(); // רשימת כל השעות הפנויות
        Node<int> 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 PrintAvailableDay (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
                Console.WriteLine (p.GetInfo().GetDay());
                Console.WriteLine (p.GetInfo().GetMonth());
            }
            p = p.GetNext();
        }
    }
}

```

# פתרון סדרק עני - מוצגים היילובים 2009

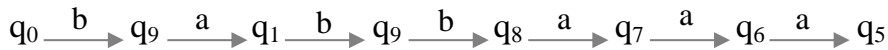
## פתרון שאלה 13

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

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

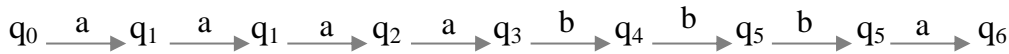
i. המילים הבאות מתקבלות/לא מתקבלות על-ידי האוטומט A:

(1) המילה babbaaa בשפה. להלן המסלול המקבל:



(2) המילה aababaaa לא בשפה.

(3) המילה aaaabbbba בשפה. להלן המסלול המקבל:



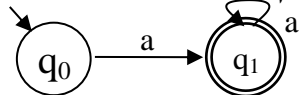
ii. המילים הקצרות ביותר המתקבלות על-ידי האוטומט A הן: aaabb, bbaaa.

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

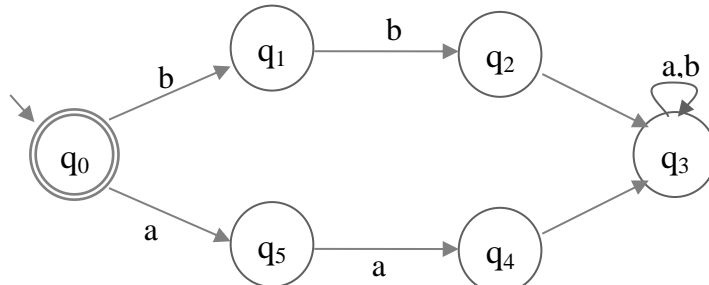
נושא מרכזי: אוטומט סופי דטרמיניסטי  
סוג השאלה: השלמת אוטומט

נתוני השאלה:

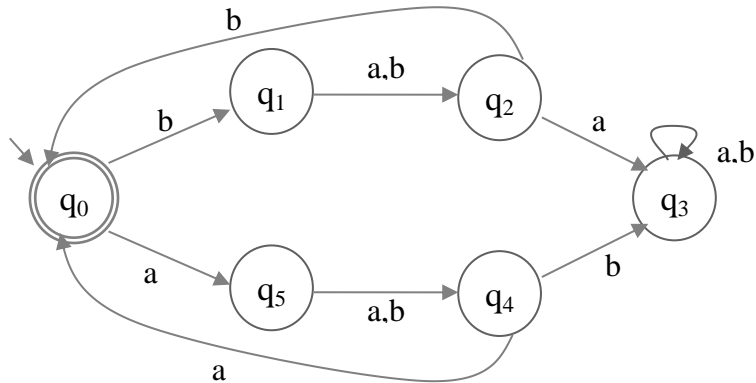
מילה באורך 3 תיקרא פלינדרום באורך 3, אם התו הראשון במילה שווה לתו האחרון במילה.

נתונה השפה L הבאה:  $L = \{w \mid w \text{ היא שרשרת של } 0 \text{ או יותר פלינדרומים באורך } 3\}$

נתון ציור חלקי של אוטומט סופי דטרמיניסטי המקבל את השפה L.



א. האוטומט המלא שמקבל את השפה  $L$ :



לא נדרש  
אך חשוב



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

נימוק: השפה  $(aab)^n$  בנויה משלשות 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:  $0^n 1^n \mid n \geq 1$

ב. השפה  $\text{Init}(L_3)$  היא:

$$\text{Init}(L_3) = \{u \mid uv \in L_3, u, v \in \Sigma^*\} =$$

$$\{0^i 1^{i-j} \cdot 1^j \mid i \geq 0, 0 \leq j \leq i\}$$

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

הסבר:  $\text{Init}(L_3)$  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,

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

$$v = 1^j \mid j \geq 0$$

ג. השפה  $\text{Fin}(L_3)$  היא:  $1^j$

הסבר – ראה סעיף ב לעיל.

ד.  $0011 \notin \text{Min}(L_4)$

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



$\{ \varepsilon \cdot 0011, 0 \cdot 011, 00 \cdot 11, 001 \cdot 1 \}$

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

ה.  $0011 \notin \text{Min}(L_5)$

נימוק: (בדומה לסעיף ד לעיל)

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

השרשורים היוצרים את המילה 0011 כאשר  $w_2 \notin \varepsilon$  הם:  $\{ \varepsilon \cdot 0011, 0 \cdot 011, 00 \cdot 11, 001 \cdot 1 \}$ . כל אחת מההתחלות (רישות) שייכת לשפה  $L_5$  ועל-כן המילה 0011 אינה שייכת ל-  $\text{Min}(L_5)$ .

ו. השפה  $L_4 \cap L_5$  אינה רגולרית.

$$L_4 \cap L_5 = \{ 0^i 1^k \mid 0 \leq i \leq k \} \cap \{ 0^i 1^k \mid 0 \leq k \leq i \} = \{ 0^i 1^i \mid i \geq 0 \}$$

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.