

מדעי המחשב

פרק ראשון

שאלה 1

Java

```

//--- פעלה המחזירה אמת אם המחרוזת המתקבלת היא מחרוזת כפולה ושקר אחרת ---
//--- מחרוזת כפולה היא מחרוזת לא ריקה שהחצי הראשון שלה זהה לחצי השני ---
public static boolean isDouble(String str)
{
    int k = str.length();
    if (k == 0)
        return false;
    String st1 = str.substring(0, k/2);
    String st2 = str.substring(k/2);
    return st1.equals(st2);
}

```

C#

```

//--- פעלה המחזירה אמת אם המחרוזת המתקבלת היא מחרוזת כפולה ושקר אחרת ---
//--- מחרוזת כפולה היא מחרוזת לא ריקה שהחצי הראשון שלה זהה לחצי השני ---
public static bool IsDouble(string str)
{
    int k = str.Length;
    if (k == 0)
        return false;
    string st1 = str.Substring(0, k / 2);
    string st2 = str.Substring(k / 2);
    return st1.Equals(st2);
}

```

שאלה 2

Java

```

//--- פעולה המחזירה מערך של זוגות מספרים אקראיים ---
public static PairOfNums[] generate(int n)
{
    PairOfNums[] arr = new PairOfNums[n];

    for (int i = 0 ; i < arr.length ; i++)
        arr[i] = getPairOfNum();

    return arr;
}

static Random rnd = new Random();

//--- פעולה המחזירה עצם מסוג התחלה-סוף תקין ---
public static PairOfNums getPairOfNum()
{
    int n1, n2;
    PairOfNums pon = new PairOfNums(-1, -1);

    do{
        n1 = rnd.nextInt( bound: 1000)+1;
        n2 = rnd.nextInt( bound: 1000)+1;
        pon = new PairOfNums(n1, n2);
    }while (!pon.endStart());

    return pon;
}

```

C#

```

//--- פעולה המחזירה מערך של זוגות מספרים אקראיים ---
public static PairOfNums[] Generate(int n)
{
    PairOfNums[] arr = new PairOfNums[n];

    for (int i = 0; i < arr.Length; i++)
        arr[i] = GetPairOfNum();

    return arr;
}

static Random rnd = new Random();

//--- פעולה המחזירה עצם מסוג התחלה-סוף תקין ---
public static PairOfNums GetPairOfNum()
{
    int n1, n2;
    PairOfNums pon = new PairOfNums(-1, -1);

    do
    {
        n1 = rnd.Next(1,1001);
        n2 = rnd.Next(1,1001);
        pon = new PairOfNums(n1, n2);
    } while (!pon.EndStart());

    return pon;
}

```

שאלה 3

Java

```

//--- פעולה המחזירה אמת אם המטריצה היא מערך פינה ושקר אחרת ---
//--- מערך פינה הוא מטריצה שבו בכל שורה ובכל עמודה תואמת ---
//--- קיים מספר השווה למס' השורה/עמודה + 1 ---
public static boolean isCorner (int[][]matrix)
{
    int col = matrix[0].length;
    int row = matrix.length;
    if (col != row)
        return false;

    for (int i = 0 ; i < row ; i++)
        if (!chkRow(matrix, i, num: i+1) || !chkCol(matrix, i, num: i+1))
            return false;

    return true;
}

//--- פעולה המקבלת את המטריצה, מספר aury ומספר שלם ---
//--- ומחזירה אמת אם כל הערכים בשורה שווים למספר ושקר אחרת ---
public static boolean chkRow(int[][]matrix, int r, int num)
{
    int lastCol = matrix[r].length;

    for (int i = r ; i < lastCol ; i++)
        if (matrix[r][i] != num)
            return false;

    return true;
}

//--- פעולה המקבלת את המטריצה, מספר העמודה ומספר שלם ---
//--- ומחזירה אמת אם כל הערכים בעמודה שווים למספר ושקר אחרת ---
public static boolean chkCol(int[][]matrix, int c, int num)
{
    int lastRow = matrix.length;

    for (int i = c ; i < lastRow ; i++)
        if (matrix[i][c] != num)
            return false;

    return true;
}

```

Java

פתרון של דפנה לוי-רשתי

```
public static boolean isCorner(int[][] mat) {
    for(int i = 0; i < mat.length; i++) {
        if (mat.length != mat[i].length) return false;
        for(int j = 0; j < mat.length; j++) {
            if (i <= j) {
                if (mat[i][j] != i+1) return false;
            }
            else
                if (mat[i][j] != j+1) return false;
        }
    }
    return true;
}
```

C#

פתרון של אחמד כריים

```
public static bool IsCorner(int[,] a)
{
    for (int i = 0; i < a.GetLength(0); i++)
        for (int j = i; j < a.GetLength(1); j++)
            if (a[i, j] != i + 1 || a[i, j] != a[j, i])
                return false;
    return true;
}
```

C#

```

//--- פעולה המחזירה אמת אם המטריצה היא מערך פינה ושקר אחרת ---
//--- מערך פינה הוא מטריצה שבו בכל שורה ובכל עמודה תואמת ---
//--- קיים מספר השווה למס' השורה/עמודה + 1 ---
public static bool IsCorner(int[,] matrix)
{
    int col = matrix.GetLength(1);
    int row = matrix.GetLength(0);
    if (col != row)
        return false;

    for (int i = 0; i < row; i++)
        if (!ChkRow(matrix, i, i + 1) || !ChkCol(matrix, i, i + 1))
            return false;

    return true;
}

//--- ומספר שלם aurv פעולה המקבלת את המטריצה, מספר ---
//--- ומחזירה אמת אם כל הערכים בשורה שווים למספר ושקר אחרת ---
public static bool ChkRow(int[,] matrix, int r, int num)
{
    int lastCol = matrix.GetLength(1);

    for (int i = r; i < lastCol; i++)
        if (matrix[r,i] != num)
            return false;

    return true;
}

//--- פעולה המקבלת את המטריצה, מספר העמודה ומספר שלם ---
//--- ומחזירה אמת אם כל הערכים בעמודה שווים למספר ושקר אחרת ---
public static bool ChkCol(int[,] matrix, int c, int num)
{
    int lastRow = matrix.GetLength(0);

    for (int i = c; i < lastRow; i++)
        if (matrix[i,c] != num)
            return false;

    return true;
}

```

פרק שני

עלה 4

Java

```
//--- פעולה המחזירה אמת אם הרשימה היא שרשרת מאורגנת ושקר אחרת ---
//--- שרשרת מאורגנת היא רשימה של מספרים שלמים באורך זוגי ---
//--- שכל המספרים בחצי הראשון קטנים מכל המספרים בחצי השני ---
//---
//--- כדי לבדוק את התכונה השנייה מספיק לבדוק שהערך הגדול ביותר
//--- בחצי הראשון קטן מהערך הקטן ביותר בחצי השני ---
public static boolean isArranged (Node<Integer>lst)
{
    int k = size(lst);
    if (k % 2 != 0)
        return false;
    int max = maxInSubList(lst, size: k/2);

    Node<Integer> pos = getPosition(lst, size: k/2);
    int min = minInSubList(pos);

    return min >= max;
}

//--- הפעולה מחזירה את מספר האיברים ברשימה ---
public static int size(Node<Integer>lst)
{
    int count=0;

    while(lst != null)
    {
        count++;
        lst = lst.getNext();
    }

    return count;
}

//--- פעולה המחזירה את המקום הפנימי לחוליה במקום הנתון ברשימה ---
public static Node<Integer> getPosition (Node<Integer> lst, int size)
{
    while (lst != null && size > 0)
    {
        lst = lst.getNext();
        size--;
    }

    return lst;
}

//--- פעולה המחזירה את המספר הגדול ביותר ברשימה באורך הנתון ---
public static int maxInSubList(Node<Integer> lst, int size)
{
    int max = lst.getValue();
    lst = lst.getNext();
    size--;

    while (lst != null && size > 0)
    {
        if (lst.getValue() > max)
            max = lst.getValue();
        lst = lst.getNext();
        size--;
    }

    return max;
}

//--- פעולה המחזירה את המספר הקטן ביותר ברשימה החל מהמקום הנתון ---
public static int minInSubList(Node<Integer>pos)
{
    int min = pos.getValue();
    pos = pos.getNext();

    while (pos != null)
    {
        if (pos.getValue() < min)
            min = pos.getValue();
        pos = pos.getNext();
    }

    return min;
}
```

יעילות הפעולה: $O(n)$

n מייצג את מספר האיברים ברשימה

הפעולות:

אורך רשימה - $size$

מקום ברשימה - $getPosition$

מקסימום ברשימה - $maxInSubList$

מינימום ברשימה - $minInSubList$

עוברות כל אחת פעם אחת על הרשימה ולכן יעילותן $O(n)$

הפעולה $isArranged$ מפעילה את כל אחת מהפעולות פעם

אחת בלבד, ולכן: $f(n) = 4n$

וסיבוכיות הפעולה: $O(n)$

C#

```
//--- פעולה המחזירה אמת אם הרשימה היא שרשרת מאורגנת ושקר אחרת ---
//--- שרשרת מאורגנת היא רשימה של מספרים שלמים באורך זוגי ---
//--- שכל המספרים בחצי הראשון קטנים מכל המספרים בחצי השני ---
//---
//--- כדי לבדוק את התכונה השנייה מספיק לבדוק שהערך הגדול ביותר ---
//--- בחצי הראשון קטן מהערך הקטן ביותר בחצי השני ---
public static bool IsArranged(Node<int> lst)
{
    int k = Size(lst);
    if (k % 2 != 0)
        return false;
    int max = MaxInSubList(lst, k / 2);

    Node<int> pos = GetPosition(lst, k / 2);
    int min = MinInSubList(pos);

    return min >= max;
}

//--- הפעולה מחזירה את מספר האיברים ברשימה ---
public static int Size(Node<int> lst)
{
    int count = 0;

    while (lst != null)
    {
        count++;
        lst = lst.GetNext();
    }

    return count;
}

//--- פעולה המחזירה הפנייה לחוליה במקום הנתון ברשימה ---
public static Node<int> GetPosition(Node<int> lst, int size)
{
    while (lst != null && size > 0)
    {
        lst = lst.GetNext();
        size--;
    }
    return lst;
}

//--- פעולה המחזירה את המספר הגדול ביותר ברשימה באורך הנתון ---
public static int MaxInSubList(Node<int> lst, int size)
{
    int max = lst.GetValue();
    lst = lst.GetNext();
    size--;

    while (lst != null && size > 0)
    {
        if (lst.GetValue() > max)
            max = lst.GetValue();
        lst = lst.GetNext();
        size--;
    }
    return max;
}

//--- פעולה המחזירה את המספר הקטן ביותר ברשימה החל מהמקום הנתון ---
public static int MinInSubList(Node<int> pos)
{
    int min = pos.GetValue();
    pos = pos.GetNext();

    while (pos != null)
    {
        if (pos.GetValue() < min)
            min = pos.GetValue();
        pos = pos.GetNext();
    }
    return min;
}
```

יעילות הפעולה: $O(n)$

n מייצג את מספר האיברים ברשימה

הפעולות:

אורך רשימה - Size

מקום ברשימה - GetPosition

מקסימום ברשימה - MaxInSubList

מינימום ברשימה - MinInSubList

עוברות כל אחת פעם אחת על הרשימה ולכן יעילותן $O(n)$

הפעולה IsArrange מפעילה את כל אחת מהפעולות פעם

אחת בלבד, ולכן: $f(n) = 4n$

וסיבוכיות הפעולה: $O(n)$

פתרון אחר (פחות יעיל) *efale* 4

Java

```

//--- פעולה המחזירה אמת אם הרשימה היא שרשרת מאורגנת ושקר אחרת ---
//--- שרשרת מאורגנת היא רשימה של מספרים שלמים באורך זוגי ---
//--- שכל המספרים בחצי הראשון קטנים מכל המספרים בחצי השני ---
public static boolean isArranged(Node<Integer> lst)
{
    int k = Size(lst);
    if (k % 2 != 0)
        return false;

    k = k / 2;

    //--- העתקת החצי השמאלי של הרשימה המקורית לרשימה חדשה ---
    Node<Integer> lst1 = new Node<Integer> (lst.getValue());
    Node<Integer> pos1 = lst1;
    Node<Integer> pos = lst.getNext();
    for (int i = 1; i < k; i++)
    {
        pos1.setNext(new Node<Integer>(pos.getValue()));
        pos1 = pos1.getNext();
        pos = pos.getNext();
    }

    //--- העתקת החצי הימני של הרשימה המקורית לרשימה חדשה ---
    Node<Integer> lst2 = new Node<Integer>(pos.getValue());
    Node<Integer> pos2 = lst2;
    pos = pos.getNext();
    while(pos != null)
    {
        pos2.setNext(new Node<Integer>(pos.getValue()));
        pos2 = pos2.getNext();
        pos = pos.getNext();
    }

    //--- האם כל איברי תת-רשימה 1 קטנים מכל איברי תת-רשימה 2 ---
    pos1 = lst1;
    while (pos1 != null)
    {
        pos2 = lst2;
        while (pos2 != null)
        {
            if (pos1.getValue() >= pos2.getValue())
                return false;
            pos2 = pos2.getNext();
        }
        pos1 = pos1.getNext();
    }

    return true;
}

```

```

//--- הפעולה מחזירה את מספר האיברים ברשימה ---
public static int Size(Node<Integer> lst)
{
    int count = 0;

    while (lst != null)
    {
        count++;
        lst = lst.getNext();
    }

    return count;
}

```

יעילות הפעולה: $O(n^2)$

n מייצג את מספר האיברים ברשימה

סיבוכיות הפעולה אורך רשימה - size הוא $O(n)$

הפעולה isArranged מחשבת:

העתקת חצי הרשימה הימני והשמאלי לרשימות חדשות, כל לולאה עוברת על מחצית הרשימה, ולכן בסה"כ $O(n)$

לאחר מכן משווה כל אחד מאיברי תת-רשימה-1 ($\frac{1}{2}n$) איברים) מול כל אחד מאיברי תת-רשימה-2 ($\frac{1}{2}n$ איברים), סה"כ $\frac{1}{2}n^2$ צעדים

$$f(n) = \frac{1}{2}n^2 + 2n \Rightarrow O(n^2)$$

C#

```

//--- פעולה המחזירה אמת אם הרשימה היא שרשרת מאורגנת ושקר אחרת ---
//--- שרשרת מאורגנת היא רשימה של מספרים שלמים באורך זוגי ---
//--- שכל המספרים בחצי הראשון קטנים מכל המספרים בחצי השני ---
public static bool IsArranged(Node<int> lst)
{
    int k = Size(lst);
    if (k % 2 != 0)
        return false;

    k = k / 2;

    //--- העתקת החצי השמאלי של הרשימה המקורית לרשימה חדשה ---
    Node<int> lst1 = new Node<int> (lst.GetValue());
    Node<int> pos1 = lst1;
    Node<int> pos = lst.GetNext();
    for (int i = 1; i < k; i++)
    {
        pos1.SetNext(new Node<int>(pos.GetValue()));
        pos1 = pos1.GetNext();
        pos = pos.GetNext();
    }

    // --- העתקת החצי הימני של הרשימה המקורית לרשימה חדשה ---
    Node<int> lst2 = new Node<int>(pos.GetValue());
    Node<int> pos2 = lst2;
    pos = pos.GetNext();
    while(pos != null)
    {
        pos2.SetNext(new Node<int>(pos.GetValue()));
        pos2 = pos2.GetNext();
        pos = pos.GetNext();
    }

    //--- האם כל איברי תת-רשימה 1 קטנים מכל איברי תת-רשימה 2 ---
    pos1 = lst1;
    while (pos1 != null)
    {
        pos2 = lst2;
        while (pos2 != null)
        {
            if (pos1.GetValue() >= pos2.GetValue())
                return false;
            pos2 = pos2.GetNext();
        }
        pos1 = pos1.GetNext();
    }

    return true;
}

//--- הפעולה מחזירה את מספר האיברים ברשימה ---
public static int Size(Node<int> lst)
{
    int count = 0;

    while (lst != null)
    {
        count++;
        lst = lst.GetNext();
    }

    return count;
}

```

יעילות הפעולה: $O(n^2)$

n מייצג את מספר האיברים ברשימה

סיבוכיות הפעולה אורך רשימה - Size הוא $O(n)$

הפעולה IsArrange מחשבת:

העתקת חצי הרשימה הימני והשמאלי לרשימות חדשות,
כל לולאה עוברת על מחצית הרשימה, ולכן בסה"כ $O(n)$

לאחר מכן משווה כל אחד מאיברי תת-רשימה-1 ($\frac{1}{2}n$)
איברים) מול כל אחד מאיברי תת-רשימה-2 ($\frac{1}{2}n$ איברים),
סה"כ $\frac{1}{2}n^2$ צעדים

$$f(n) = \frac{1}{2}n^2 + 2n \Rightarrow O(n^2)$$

Java

פתרון של דפנה לוי-רשתי

```
public boolean isArranged(Node<Integer> lst) {
    int len = len(lst);
    if(len%2 == 1) return false;
    int max = lst.getValue();
    for (int i = 0; i < len/2; i++) {
        if (max > lst.getValue())
            max = lst.getValue();
        lst = lst.getNext();
    }
    while(lst!=null) {
        if (lst.getValue() <= max) return false;
        lst = lst.getNext();
    }
    return true;
}
```

C#

פתרון של דיתה אוהב ציון

```
// פעולת עזר: פעולה המחזירה את אורך השרשרת
1 reference
public static int Size(Node<int> lst)
{
    int count = 0;
    while (lst != null)
    {
        count++;
        lst = lst.GetNext();
    }
    return count;
}

// פעולת עזר: פעולה המחזירה את האיבר במיקום מבוקש
1 reference
public static Node<int> Find(Node<int> lst, int place)
{
    for (int i = 0; i < place; i++)
        lst = lst.GetNext();
    return lst;
}

// פעולת עזר
// פעולה המקבלת שרשרת חוליות ומספר ומחזירה אמת אם המספר קטן מכל המספרים בשרשרת
1 reference
public static bool IsLessAll(Node<int> lst, int num)
{
    while (lst != null)
    {
        if (num >= lst.GetValue()) return false;
        lst = lst.GetNext();
    }
    return true;
}

// הפעולה המבוקשת בשאלה
2 references
public static bool IsArranged(Node<int> lst)
{
    int count = Size(lst);
    if (count % 2 != 0)
        return false;

    // החוליה הראשונה של החצי הגדול
    Node<int> second = Find(lst, count / 2);

    // סריקה של החצי הראשון, ושליחת כל מספר לבדיקה עם החצי השני
    for (int i = 0; i < count / 2; i++)
    {
        if (!IsLessAll(second, lst.GetValue()))
            return false;
        lst = lst.GetNext();
    }
    return true;
}
```

שאלה 5

Java

```
//--- סעיף א ---
//--- פעולה המחזירה אמת אם הרשימה הראשונה מהווה את תחילתה של הרשימה השנייה ושקר אחרת ---
public static boolean isPrefix(Node<Integer> lst1, Node<Integer> lst2)
{
    Node<Integer>pos1 = lst1;
    Node<Integer>pos2 = lst2;
    while (pos1 != null && pos2 != null)
    {
        if (pos1.getValue() != pos2.getValue())
            return false;
        pos1 = pos1.getNext();
        pos2 = pos2.getNext();
    }

    return pos1 == null;
}

//--- סעיף ב ---
//--- פעולה המחזירה אמת אם הרשימה הראשונה מוכלת בתוך הרשימה השנייה ברצף ושקר אחרת ---
public static boolean isSubChain(Node<Integer> lst1, Node<Integer> lst2)
{
    while (lst2 != null)
    {
        if (isPrefix(lst1, lst2))
            return true;
        lst2 = lst2.getNext();
    }
    return false;
}
```

C#

```
//--- סעיף א ---
//--- פעולה המחזירה אמת אם הרשימה הראשונה מהווה את תחילתה של הרשימה השנייה ושקר אחרת ---
public static bool IsPrefix(Node<int> lst1, Node<int> lst2)
{
    Node<int> pos1 = lst1;
    Node<int> pos2 = lst2;
    while (pos1 != null && pos2 != null)
    {
        if (pos1.GetValue() != pos2.GetValue())
            return false;
        pos1 = pos1.GetNext();
        pos2 = pos2.GetNext();
    }
    return pos1 == null;
}

// --- סעיף ב ---
//--- פעולה המחזירה אמת אם הרשימה הראשונה מוכלת בתוך הרשימה השנייה ושקר אחרת ---
public static bool IsSubChain(Node<int> lst1, Node<int> lst2)
{
    while (lst2 != null)
    {
        if (IsPrefix(lst1, lst2))
            return true;
        lst2 = lst2.GetNext();
    }
    return false;
}
```

שאלה 6

Java

--- פעולה המקבלת תור של נבדקים ומחזירה את קוד העיר שבה הכי הרבה חולים ---

```
public static int mostSick(Queue<CovidTest>q)
{
    int max = 0;
    int cityCode = 0;

    int num = 0;
    while (!q.isEmpty())
    {
        int code = q.head().getCityCode();
        num = numOfSickInCity(q, code);
        if (num > max)
        {
            max = num;
            cityCode = code;
        }
    }
    return cityCode;
}
```

--- פעולה המקבלת תור של חולים וקוד של עיר ומחזירה את מספר החולים מעיר זו בתור ---

--- בסיום הפעולה לא יהיו יותר נבדקים מעיר זו בתור ---

```
public static int numOfSickInCity (Queue<CovidTest> q, int code)
{
    Queue<CovidTest> qTemp = new Queue<CovidTest>();
    int count = 0;
    CovidTest ct;

    while (! q.isEmpty())
    {
        ct = q.remove();
        if (ct.getCityCode() == code && ct.isSick())
            count ++;
        else
            qTemp.insert(ct);
    }

    while (!qTemp.isEmpty())
        q.insert(qTemp.remove());

    return count;
}
```

C#

--- פעולה המקבלת תור של נבדקים ומחזירה את קוד העיר שבה הכי הרבה חולים ---

public static int MostSick(Queue<CovidTest> q)

```
{
    int max = 0;
    int cityCode = 0;

    int num = 0;
    while (!q.IsEmpty())
    {
        int code = q.Head().GetCityCode();
        num = NumOfSickInCity(q, code);
        if (num > max)
        {
            max = num;
            cityCode = code;
        }
    }
    return cityCode;
}
```

--- פעולה המקבלת תור של חולים וקוד של עיר ומחזירה את מספר החולים מעיר זו בתור ---

--- בסיום הפעולה לא יהיו יותר נבדקים מעיר זו בתור ---

public static int NumOfSickInCity(Queue<CovidTest> q, int code)

```
{
    Queue<CovidTest> qTemp = new Queue<CovidTest>();
    int count = 0;
    CovidTest ct;

    while (!q.IsEmpty())
    {
        ct = q.Remove();
        if (ct.GetCityCode() == code && ct.IsSick())
            count++;
        else
            qTemp.Insert(ct);
    }

    while (!qTemp.IsEmpty())
        q.Insert(qTemp.Remove());

    return count;
}
```

שאלה 7

א. sod1 (123)

x	x < 10	משפט זימון	ערך מוחזר
123	לא	sod1 (12)	
12	לא	sod1 (1)	
1	כן		1
הפעולה מחזירה: 1			

מטרת הפעולה: הפעולה מחזירה את הספרה המשמעותית ביותר במספר (הספרה השמאלית ביותר)

ב. sod2 (123)

x	x < 10	משפט זימון	ערך מוחזר
123	לא	sod2 (12)	$\underline{2} * 10 + 3 = 23$
12	לא	sod2 (1)	$\underline{0} * 10 + 2 = 2$
1	כן		0
הפעולה מחזירה: 23			

מטרת הפעולה: הפעולה מחזירה את המספר ללא הספרה המשמעותית ביותר שבו (ללא הספרה השמאלית ביותר)

ג. sod3 (123, 68)

x	y	y == 10	tmp1	tmp2	משפט זימון	ערך מוחזר
123	68	לא	1236	8	sod3 (1236, 8)	
12	8	לא	12368	0	sod3 (12368, 0)	
1	0	כן				12368
(1) הפעולה מחזירה: 12368						
(2) עבור sod3(35, 792) יוחזר 35792						
(3) מטרת הפעולה: להחזיר מספר שלם חדש שתחילתו המספר הראשון וסופו המספר השני ("לשרשר" את המספר השני לסופו של המספר הראשון)						

פרק פסי

לפניך שאלות מ-4 מסלולים שונים: מערכות מחשב ואסמבלי (שאלות 8-9), מבוא לחקר ביצועים (שאלות 10-11), מודלים חישוביים (שאלות 12-13), תכנות מונחה עצמים בשפת Java (שאלות 14-15), תכנות מונחה עצמים בשפת C# (שאלות 16-17).

מערכות מחשב ואסמבלי

פתרון לפרק זה נכתב ע"י: רמי דיין

עלה 8

<p style="text-align: center;"><u>א</u></p> <pre> AGAIN: cmp bx, 10 jg SOF cmp cx, 0 jle SOF mov ax, bx mov dl, 2 idiv dl cmp ah, 0 je ELSE mov ax, bx imul dl mov bx, ax jmp AGAIN ELSE: sub cx, 2 jmp AGAIN SOF: nop </pre>	<p style="text-align: center;"><u>ב</u></p> <pre> proc test1 mov bp, sp mov al, 1 mov cx, [bp+2] inc cl cmp cl, ch je SOF xor al, al SOF: ret 2 endp </pre>
---	--

עלה 9

<p style="text-align: center;"><u>ב</u></p> <ol style="list-style-type: none"> 1. בסוף הקטע $dl = 25$ 2. הערך צריך להיות 10 3. קטע הקוד מכניס ל dl את הערך ההתחלתי שלו בחזקת 2 	<p style="text-align: center;"><u>א</u></p> <pre> xor di, di lea bx, ARR1 lea si, ARR2 mov cx, 5 AGAIN: mov ax, [bx] add ax, [bx+2] add bx, 4 mov [si], ax add si, 2 loop AGAIN </pre>
---	--

מערכות מחשב ואסמבלי

פתרון לפרק זה נכתב ע"י: רן ויינשטיין

שאלה 8

שאלה 8א.

שאלה 8ב.

```
PROC TEST
    MOV AL, 0
    MOV BP, SP
    MOV CX, [BP+2]
    INC CL
    CMP CH, CL
    JNE SOF
    ADD AL, 1
    SOF:
        RET 2
ENDP TEST
```

AGAIN:

```
CMP BX, 10
JG SOF
CMP CX, 0
JL SOF
MOV AX, 1
AND AX, BX
CMP AX, 1
JNE ZOGI
SHL BX, 1
JMP AGAIN
```

ZOGI:

```
SUB CX, 2
JMP AGAIN
```

SOF:

```
NOP
```

שאלה 9

שאלה 9א.

שאלה 9ב.

1. 25
2. 10
3. חישוב של העלאה בריבוע

```
MOV CX, 5
MOV AX, 0
LEA SI, ARR1
LEA DI, ARR2
AGAIN:
    MOV AX, [SI]
    ADD AX, [SI+2]
    MOV [DI], AX
    ADD SI, 4
    ADD DI, 2
    LOOP AGAIN
```

מבוא לחקר ביצועים

עאלה 10

עאלה 11

מודלים חישוביים

נכתב ע"י: רחל לודמר

שאלה 12

א.

$$L_1 = \{ \text{כל המילים שבהן מספר ה } b \text{ גדול ממספר ה } a \}$$

$$L_2 = \{ \text{כל המילים שבהן } b \text{ מופיע לפחות 3 פעמים} \}$$

$$L_3 = L_1 \cap \overline{L_2}$$

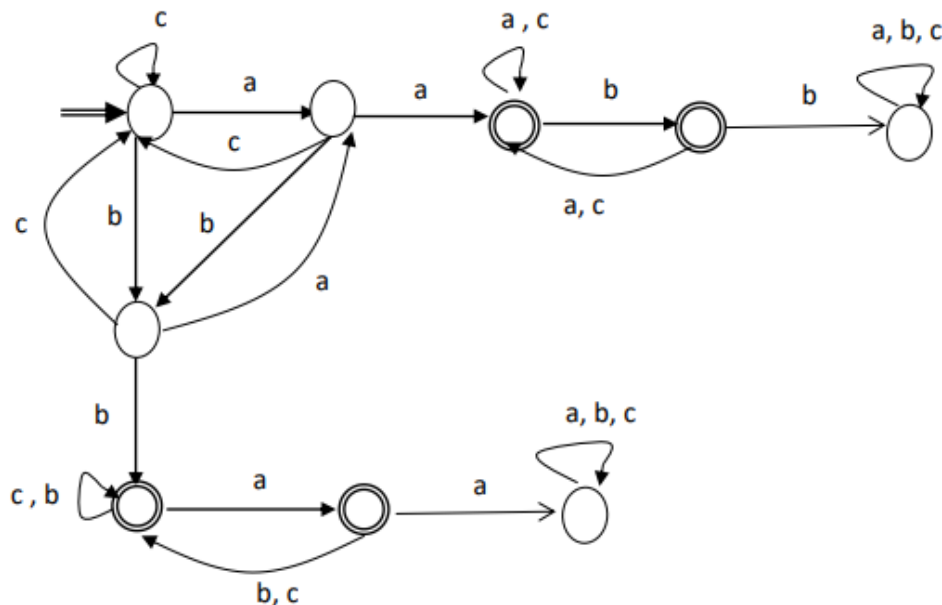
נרשום את כל המילים בשפה L_3 :

$$L_3 = \{ \text{כל המילים שבהן מספר ה } b \text{ גדול ממספר ה } a \} \cap \{ \text{כל המילים שיש בהן } b \text{ לפחות 3 פעמים} \}$$

$$L_3 = \{b, bba, aba, abb\}$$

ב.

נבנה אוטומט סופי דטרמיניסטי מלא המכיל aa או bb אך לא שניהם.



שאלה 13

$$L = \{a^n b^k = a^n b^{n/2} b^{n\%2} \mid n > 0\}$$

נתונה השפה :

א. לכל n מ 1 עד 6

n	$k=n/2+n\%2$	המילה בשפה
1	$1/2+1=0+1=1$	ab
2	$2/2+0=1$	aab
3	$3/2+1=1+1=2$	aaabb
4	$4/2+0=2$	aaaabb
5	$5/2+1=2+1=3$	aaaaabbb
6	$6/2+0=3$	aaaaaabb

ב.

רואים לפי הטבלה שעבור רצפים n ו $n+1$ (איזוגי וזוגי סמוכים) מספר ה b זהה וגם באופן מתמטי:

$$a^{2n-1} b^k = a a^{2(n-1)} b^{n-1} b \quad : \text{עבור רצף אי זוגי } 2n-1 \text{ לכל } n \geq 1$$

$$a^{2n} b^k = a^{2n} b^n \quad : \text{עבור רצף זוגי } 2n \text{ לכל } n \geq 1$$

לכן, הרעיון באוטומט המחסנית הוא: שעבור רצף a , על כל זוג נדחף סימן אחד ועל כל b נשלף.

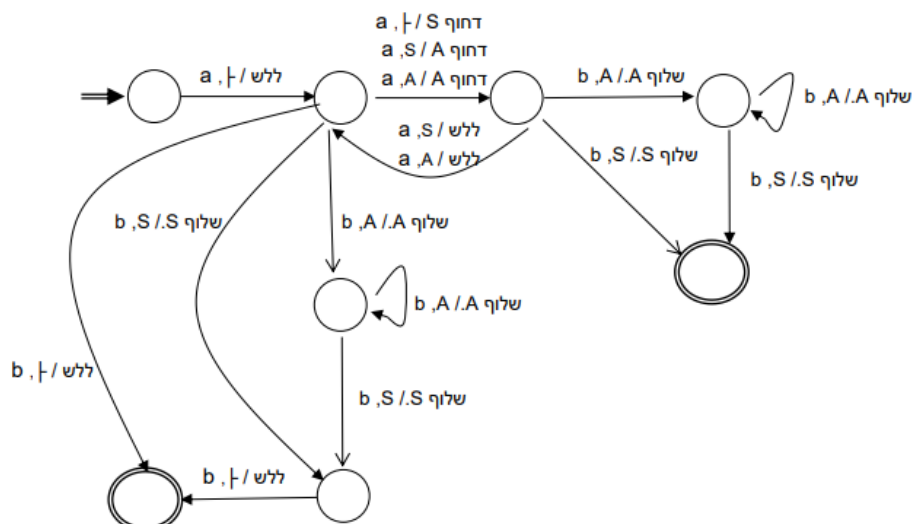
אם נתחיל בדחיפה בתו הראשון של a , אז עבור המקרה של n אי זוגי, ה b האחרון של השארית צריך לשלוף את סימן הראשון (S) שדחפנו ונמצאים במצב מקבל. (כמו במקרה של רצף זוגי)

ואם לא דוחפים ב a הראשון, בלולאת הזוגיות שתיווצר אח"כ, (דוחפים סימן, בשני לא), יוצא שברצף זוגי מספר הסימנים במחסנית יהיו מחצית n וברצף אי זוגי יהיה 1 פחות. (ראה בנוסחה המתמטית)

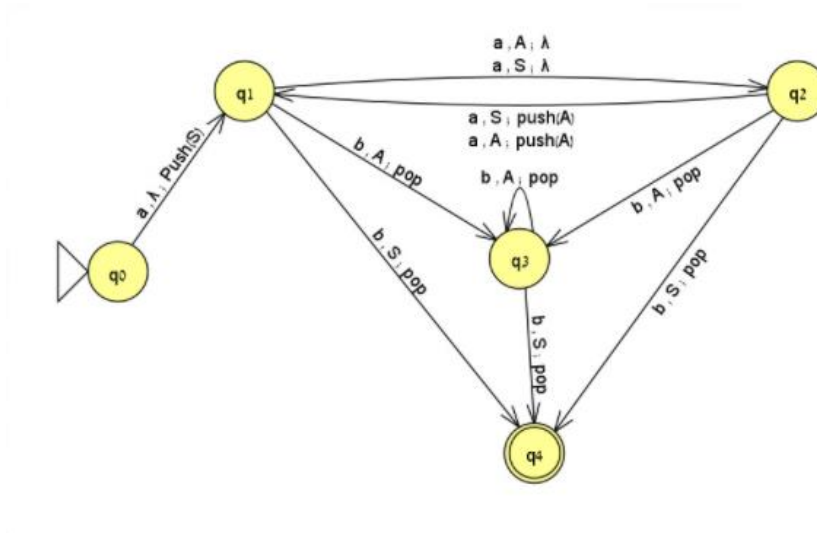
לכן, עבור רצף אי זוגי נוסיף לקרוא את ה b של השארית כאשר המחסנית ריקה..

נשים לב שבדרך זו מקיימים את המנה המתקבלת מחלוקת n (אי זוגי) ב 2.

עבור $n=1$ לא ידחף אף סימן למחסנית, ועבור $n=3$ ידחף רק סימן אחד למחסנית ($3/2=1$) וכו'....



פתרון אחר לאוטומט מחסנית, ע"י דפנה לוי-רשתי



תכנות מונחה עצמים בשפת Java

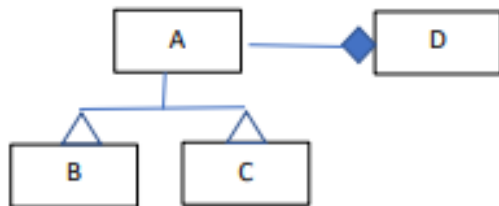
נכתב ע"י: אביטל Evi גרינולד

שאלה 14

השאלה עוסקת ב:

השלמת היררכיית מחלקות: יחסי ירושה והכלה, מעקב בעזרת תרשים עצמים.

סעיף א: היררכיית המחלקות A,B,C,D



סעיף ב: מעקב עם תרשים עצמים

1. `<A> y1` →
2. `<A> y2` →
3. `y1.foo(3)` `<A> y1` →
4. `y1.isEqual(y2)` output: **true**
5. ` y3` →
6. `<A> y4` →
7. `y3.isEqual(y4)` output: **true** [activates methods from A]
8. `y4.isEqual(y3)` output: **true** [activates methods from A]
9. ` y5` →
` y3` →
10. ` y5` →
` y3` →
11. `y3.isEqual(y5)` output: **true** [activates methods from B]
12. `<C> y6` →
13. output: **true** [activates methods from A, each C is A]
14. `<D> d1` → ← `y4 <A>`
15. `<D> d2` → ← `y6 <C>`

line	A.count
1	0 1
2	1 2
3	2
4	2
5	2 3
6	3 4
7	4
8	4
9	4
10	4
11	4
12	4 5
13	5
14	5
15	5
16	5
17	5
18	5
19	5

פלט:

```

true
true
true
true
true
5
I am B
I am A
true
false
  
```

16. output: 5
17. d1.func() output: I am B
18. d2.func() output: I am A
19. d1.isB() output: true
20. d2.isB() output: false

הסברים:

במחלקות A ו B יש שתי פעולות isEqual

במחלקה A החתימה של הפעולה: `public Boolean isEqual(A other)`

במחלקה B החתימה של הפעולה: `public Boolean isEqual(B other)`

המחלקה B יורשת מהמחלקה A ולכן יורשת את הפעולה שכתובה במחלקה העל, מחלקה A כלומר במחלקה B קיימות 2 פעולות, האחת שמקבלת הפנייה לעצם מטיפוס A והשנייה המקבלת הפנייה לעצם מטיפוס B.

מה ההבדל בין שתי הפעולות?

הפעולה הראשונה מחזירה אמת עם הערכים של שני העצמים זהים, האמת הנוכחי והעצם שמועבר כפרמטר, שקר – אחרת.

הפעולה השנייה מחזירה אמת אם ההפניה לשני העצמים היא אותה אחת, כלומר האם העצם המועבר כפרמטר הוא גם העצם הנוכחי, שקר – אחרת.

בשורה (7) `y3.isEqual(y4)` מופעלת הפעולה הראשונה.

למה? Y4 הוא מטיפוס המחלקה A, כי בשורה (6) התבצעה המרה לא מפורשת כלפי מעלה ממחלקה B למחלקה A, ולכן על פי חתימת הפעולה תופעל הפעולה זו שמקבלת הפנייה לעצם מטיפוס A. תוכן הערכים זהה ולכן מוחזר הערך true

בשורה (8) `y4.isEqual(y3)` שוב מופעלת הפעולה הראשונה. Y4 הוא מטיפוס המחלקה A ויכול להפעיל אך ורק פעולות הכתובות במחלקה זו. במחלקה זו אין את הפעולה השנייה.

בשורה (11) `y3.isEqual(y5)` מופעלת הפעולה השנייה מאחר וגם y3 וגם y5 הן מטיפוס המחלקה B. בשורה (9) התבצעה הפנייה של y5 לאותו העצם ש y3 מפנה אליו, לכן למעשה יש גאן עצם אחד בלבד והערך המוחזר מהפעולה השנייה הוא true.

שאלה 15

השאלה עוסקת ב:

א- ירושה, זיהוי שגיאות ופלט.

ב- השלמת קוד שייתן פלט נתון

סעיף א (1)

< C > cd → D
< D > dd →

פלט	תקין / לא תקין	הוראה	
x	תקין. למחלקה D יש פעולה dolt שמקבלת פרמטר מטיפוס C	dd.dolt(cd);	1
b	תקין. למחלקה C יש פעולה dolt שמקבלת פרמטר מטיפוס D והמחלקה D דורסת אותה, כלומר הפעולה שתבצע היא זו שכתובה במחלקה D	cd.dolt(dd);	2
	שגיאת תחביר. למחלקה C אין פעולה dolt שמקבלת פרמטר מטיפוס C	cd.dolt(cd);	3

סעיף א (2)

פלט	תקין / לא תקין	הוראה	
x	תקין. למחלקה D יש פעולה dolt שמקבלת פרמטר מטיפוס C	dd.dolt(cd);	1
b	תקין. למחלקה C יש פעולה dolt שמקבלת פרמטר מטיפוס D והמחלקה D דורסת אותה, כלומר הפעולה שתבצע היא זו שכתובה במחלקה D	cd.dolt(dd);	2
o	תקין. הפעם מאחר למחלקה C יש פעולה dolt שמקבלת עצם מטיפוס Object וכל המחלקות יורשות ממנה, תזומן פעולה זו	cd.dolt(cd);	3

סעיף ב

בשלושת הסעיפים חייבים ליצור עצם מטיפוס BB שכן בפלטים מופיע In BB והוראה כזו יש רק בפעולות הבונות במחלקה BB

הוראה	הסבר	פלט רצוי
1 BB b1 = new BB();	הפעולה הבונה הריקה מזמנת את ה בנאי הריק super() גם אם לא נכתב במפורש לכן בשורה ראשונה ייכתב In AA ובשורה שנייה הוראת ההדפסה שמופיעה בפעולה הבונה הריקה של BB	In AA In BB
2 BB b2 = new BB(7.36);	שורת הדפסה שנייה מכוונת לכך שיש צורך לזמן את הפעולה הבונה עם הפרמטר מטיפוס ממשי. ערך הפרמטר חייב להיות מספר עשרוני שהחלק הממשי הוא 7. בשורה הראשונה של בנאי מזומן הבנאי הריק של מחלקת העל, AA	In AA In BB7
3 BB b3 = new BB(3);	שתי שורות אחרונות בפלט מכוונות לכך שיש לזמן פעולה בונה ממחלקה BB אשר יש לה פרמטר שלם והוא חייב להיות 3 כדי שהדפסת In BB תופיע פעמיים. הפעולה הבונה מזמנת את הבנאי עם הערך 6 הפעולה הבונה של מחלקה AA עם פרמטר שלם, מזמנת את הבנאי הריק ולכן תופיע הדפסה In AA, לאחריו יש עוד הדפסה של In AA ולאחריו פעמיים 6 שזה 12	In AA In AA 12 In BB In BB

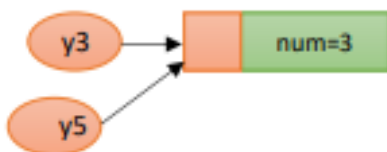

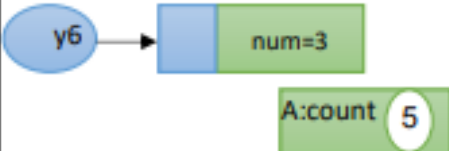

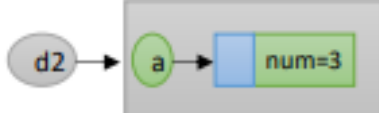
תכנות מונחה עצמים בשפת C#

נכתב ע"י: דיתה אוהב ציון

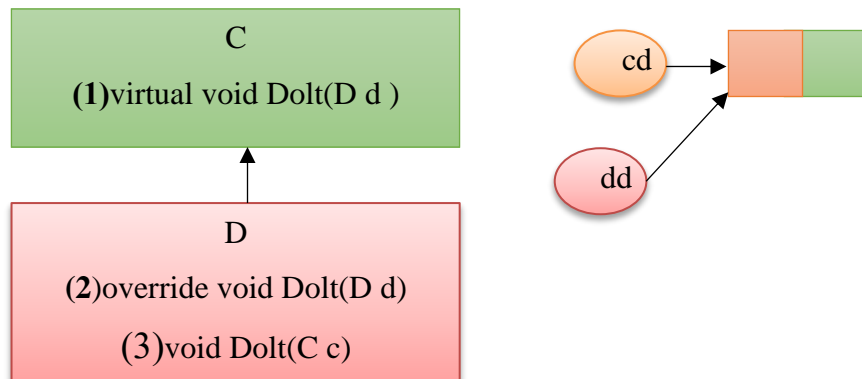
שאלה 16



<code>A y1 = new A();</code>		
<code>A y2 = new A();</code>		
<code>y1.Foo(y1.GetNum());</code>	<p>בפעולה Foo : num++ -מתייחס לפרמטר ולא לתכונה (כדי להתייחס לתכונה יש להשתמש ב this)</p>	
<code>Console.WriteLine(y1.IsEqual(y2));</code>		True
<code>B y3 = new B();</code>		
<code>A y4 = new B();</code>		
<code>Console.WriteLine(y3.IsEqual(y4));</code>	<p>במחלקה B יש שתי פעולות IsEqual - אחת התקבלה הירושה והשניה נכתבה המחלקה. השונות בטיפוס הפרמטר שהן מקבלות. מופעלת הפעולה המקבלת טיפוס A כפרמטר (פעולה ממחלקה A)</p>	True
<code>Console.WriteLine(y4.IsEqual(y3));</code>	<p>מופעלת הפעולה במחלקה A - המוכרת ע"י y4 - הפרמטר מטיפוס A יכול להחזיק את y3</p>	true

B y5 = y3;		
y5.SetNum(0);		
Console.WriteLine(y3.IsEqual(y5));	מופעלת הפעולה IsEqual במחלקה B. וכמובן תוכן שתי ההפניות זהה- מצביעות על אותו עצם.	True
C y6 = new C();		
Console.WriteLine(((B)y4).IsEqual(y6));	המרה זמנית של y4 ל B. במחלקה B יש שתי פעולות IsEqual- אחת התקבלה הירושה והשניה נכתבה המחלקה- השונות בטיפוס הפרמטר שהן מקבלות. מופעלת הפעולה המקבלת טיפוס A כפרמטר (פעולה ממחלקה A). מאחר והפרמטר מטיפוס C רק הפעולה במחלקה A יכולה לקבל אותו.	True
D d1 = new D(y4);		
D d2 = new D(y6);		
Console.WriteLine(B.count);	פניה למשתנה סטטי היא באמצעות שם המחלקה. מאחר ומחלקה B יורשת את A, אפשר לפנות לcount עם B (או C ...)	5
d1.Func();	הפעולה Func() היא וירטואלית לכן מופעלת הפעולה של המחלקה המוחזקת בפועל. הפניה מטיפוס A מחזיקה B לכן תופעל הפעולה ב B	I am B
d2.Func();	למחלקה C אין מימוש לכן מופעלת הפעולה של A.	I am A
d1.IsB();		True
d2.IsB();		False

שאלה 17



א.

1.

dd.Dolt(cd);	הפלט : X. הסבר: מופעלת פעולה 3 מאחר והפרמטר הנשלח היא מטיפוס C
cd.Dolt(dd);	הפלט : b. מופעלת פעולה 2. הפניה מטיפוס C המחזיקה D מופעלת פעולה של העצם המוחזק בפועל – כלומר overridden במחלקה D.
cd.Dolt(cd);	שגיאה. אין פעולה override המקבלת עצם מטיפוס C. הפניה cd מטיפוס C לא מכירה את פעולה 3 במחלקה D.

2. במחלקה C נוספה פעולה `public void Dolt(Object o)`

dd.Dolt(cd);	הפלט : X. הסבר: מופעלת פעולה 3 מאחר והפרמטר הנשלח היא מטיפוס C
cd.Dolt(dd);	הפלט : b. מופעלת פעולה 2. הפניה מטיפוס C המחזיקה D מופעלת פעולה של העצם המוחזק בפועל – כלומר overridden במחלקה D.
cd.Dolt(cd);	הפלט : o. מופעלת הפעולה שנוספה. הפרמטר מטיפוס Object יכול להחזיק עצם מטיפוס C.

ב.

הפעולה	הפלט
AA a = new BB();	In AA In BB
AA a = new BB(7.0);	In AA In BB7
AA a = new BB(3);	In AA In AA 12 In BB In BB