מדעי המחשב ב'

פתרון בחינת הבגרות

פרק א - עיצוב תכנה

שאלה 1:

פתרון בשפת Java:

- מציאת הזוג הראשון במחסנית 1 הגדול מהזוג המקסימאלי במחסנית 2.

אם הוא גדול מהזוג בעל הסכום המקסימאלי, הוא בהכרח גדול מסכום כל הזוגות הסמוכים במחסנית 2.
יעילות הפתרון היא O(n)

```
//---  פעולה המחזירה את סכום זוג האיברים הסמוכים ---
//---                    הגדול ביותר במחסנית ---
//---          הנחה: במחסנית יש לפחות שני איברים ---
public static int maxSumCuple(Stack<Integer> st2)
{
     int x = st2.pop();
     int y = st2.pop();
     int sum = x + y;
     while (!st2.isEmpty())
     {
          x = y;
          y = st2.pop();

          if (x + y > sum)
               sum = x + y;
     }
     return sum;
}
//---  פעולה המחזירה את סכום זוג האיברים הסמוכים הראשון ---
//---         הגדול ביותר במחסנית1 הגדול מסכום כל זוגות ---
//---                  המספרים הסמוכים במחסנית השנייה ---
//---               הנחה: בכל מחסנית יש לפחות 2 איברים ---
public static int maxCupleSt1 (Stack<Integer> st1,
                               Stack<Integer> st2)
{
     int x = st1.pop();
     int y = st1.pop();
     int sum = maxSumCupple(st2);

     while (!st1.isEmpty() && (x+y) < sum )
     {
          x = y;
          y = st1.pop();
     }
     if (st1.isEmpty())
          return 0;
     return x+y;

}
```

- בדיקת הסכום של כל זוג מספרים ממחסנית 1 מול כל זוגות המספרים במחסנית 2, עד אשר יימצא הזוג הראשון במחסנית 1 הגדול מכל הזוגות שבמחסנית 2.
יעילותו של פתרון זה היא $O(n^2)$

```
//---  פעולה המחזירה את סכום זוג האיברים הסמוכים הראשון ---
//---         הגדול ביותר במחסנית1 הגדול מסכום כל זוגות ---
//---                  המספרים הסמוכים במחסנית השנייה ---
//---               הנחה: בכל מחסנית יש לפחות 2 איברים ---
public static int maxCupleSt1 (Stack<Integer> st1,
                               Stack<Integer> st2)
{
     Stack<Integer> stTemp = new Stack<Integer>();
     int x1 = st1.pop();
     int y1 = st1.pop();
     int sum1 = x1 + y1;

     boolean bigger = true;
     while (!st1.isEmpty() && bigger)
     {
          int x2 = st2.pop();
          int y2 = st2.pop();
          stTemp.push(x2);
          stTemp.push(y2);

          int sum2 = x2 + y2;
          if (sum1 <= sum2)
               bigger = false;

          while (!st2.isEmpty() && bigger)
          {
               x2 = y2;
               y2 = st2.pop();
               stTemp.push(y2);
               sum2 = x2 + y2;
               if (sum1 <= sum2)
                    bigger = false;
          }
          if (bigger)
               return sum1;

          while (! stTemp.isEmpty())
                    st2.push(stTemp.pop());
          bigger = true;
          x1 = y1;
          y1 = st1.pop();
          sum1 = x1 + y1;
     }
     return 0;

}
```

פתרון בשפת C# - נכתב ע"י ראמי גבאלי:

- מציאת הזוג הראשון במחסנית 1 הגדול מהזוג המקסימאלי במחסנית 2.
אם הוא גדול מהזוג בעל הסכום המקסימאלי, הוא בהכרח גדול מסכום כל הזוגות הסמוכים במחסנית 2.
יעילות הפתרון היא O(n)

```
//  פעולה מחזירה את זוג האיברים הסמוכים הגדול ביותר
public static int MaxTwo(Stack<int> s)
{
      Stack<int> s2 = new Stack<int>();
      int max = 0;
      int x, y;
      while (!s.IsEmpty())
      {
            x = s.Pop();
            s2.Push(x);
            if (!s.IsEmpty())
            {
                  y = s.Top();
                  if (x + y > max)
                        max = x + y;
            }
      }
      while (!s2.IsEmpty())
            s.Push(s2.Pop());
      return max;
}

// פעולה מחזירה את סכום של זוג האיברים הסמוכים
public static int SumB(Stack<int> st1,Stack<int> st2)
{
      int x, y, m;
      while (!st1.IsEmpty())
      {
            x = st1.Pop();
            if (!st1.IsEmpty())
            {
                  y = st1.Top();
                  m=MaxTwo(st2);
                  if (x + y > m)
                        return x + y;
            }
      }
      return 0;
}
```

- פתרון רקורסיבי: **נכתב ע"י ראמי ג'באלי**

```
/*    בוליאני ומשתנה מספרים שני, מחסנית המקבלת פעולה
 *    זוג מכל גדול המספרים זוג סכום אם "אמת" ומחזירה
 *          אחרת "שקר"-ו, אחרת במחסנית סמוכים איברים
 */
public static bool IsBig2(Stack<int> s, int x, int y, bool flag2)
{
     bool flag;
     if (s.IsEmpty() && flag2 == false)
          flag = true;
     else
          if (s.IsEmpty() && flag2)
               flag = false;
          else
          {
               int a = s.Pop();
               if (!s.IsEmpty())
               {
                    int b = s.Top();
                    if (a + b >= x + y)
                    {
                         flag = false;
                         flag2 = true;
                    }
               }
               flag = IsBig2(s, x, y, flag2);
               s.Push(a);
          }
          return flag;
     }

     // --- הסמוכים האיברים זוג הסכום את מחזירה פעולה
     public static int Sum2(Stack<int> s1, Stack<int> s2)
     {
          if (s1.IsEmpty())
               return 0;
          else
          {
               int x = s1.Pop();
               if (!s1.IsEmpty())
               {
                    int y = s1.Top();
                    if (IsBig2(s2, x, y, false))
                    return x + y;
               }
               return Sum2(s1, s2);
          }
     }
```

נכתב ע"י ראמי ג'באלי:

- בדיקת הסכום של כל זוג מספרים ממחסנית 1 מול כל זוגות המספרים במחסנית 2, עד אשר יימצא הזוג הראשון במחסנית 1 הגדול מכל הזוגות שבמחסנית 2.
יעילותו של פתרון זה היא $O(n^2)$

```
public static bool IsBig(Stack<int> s, int x, int y)
{
      Stack<int> s2 = new Stack<int>();
      bool flag = true;
      int a, b;
      while (!s.IsEmpty())
      {
            a = s.Pop();
            if (!s.IsEmpty())
            {
                  b = s.Top();
                  if (x + y <= a + b)
                        flag = false;
            }
            s2.Push(a);
      }
      while (!s2.IsEmpty())
            s.Push(s2.Pop());
      return flag;
}

//--- פעולה מחזירה את סכום של זוג האיברים הסמוכים ---
public static int Sum(Stack<int> st1, Stack<int> st2)
{
      int x, y;
      while (!st1.IsEmpty())
      {
            x = st1.Pop();
            if (!st1.IsEmpty())
            {
                  y = st1.Top();
                  if (IsBig(st2, x, y) == true)
                        return x + y;
            }
      }
      return 0;
}
```

שאלה 2:

פתרון בשפת Java:

```
//---          ט.כניסה: רשימת מספרים ---
//--- ט.יציאה: מוחזרת רשימת הטווחים ---//
public static List<RangeNode> createRangeNode
                                (List<Integer> sourceList)
{
     int from, to;
     List<RangeNode>rangeList = new List<RangeNode>();
     Node<RangeNode> pos2 = null;

     Node<Integer>pos1 = sourceList.getFirst();
     while (pos1 != null)
     {
          from = pos1.getInfo();
          to = from;
          pos1 = pos1.getNext();

          while (pos1 != null && pos1.getInfo()-1 == to)
          {
               to = pos1.getInfo();
               pos1 = pos1.getNext();
          }
          RangeNode rn = new RangeNode(from, to);
          pos2 = rangeList.insert(pos2, rn);
     }

     return rangeList;
}
```

פתרון בשפת C# - נכתב ע"י ראמי גבאלי:

```
//הטווחים רשימת מחזירה פעולה
public static List<RangeNode> CreateRangeList
                                    (List<int> sourceList)
{
     Node<int> p = sourceList.GetFirst();
     List<RangeNode> list=new List<RangeNode>();
     Node<RangeNode> q = list.GetFirst();
     int x, y=0, z;

     z = p.GetInfo();
     if (p.GetNext() == null)  //אם הרשימה מורכבת מאיבר אחד
          q = list.Insert(q, new RangeNode(z,z));
     else
     {
          while (p.GetNext() != null)
          {
               x = p.GetInfo();
               y = p.GetNext().GetInfo();
               if (y - x == 1)
               {
                    p = p.GetNext();
               }
               else
               {
                    q = list.Insert(q, new RangeNode(z, x));
                    z = y;
                    p = p.GetNext();
                }
          }
          q = list.Insert(q, new RangeNode(z, y));
     }
     return list;
}
```

פתרון בשפת C# - נכתב ע"י דורון כהן, תלמיד בתיכון מכבים-רעות:

```
public static List<RangeNode> CreateRangeList
                                (List<int> sourceList)
{
    //     הפעולה מקבלת רשימה של מספרים שלמים ומחזירה את רשימת הטווחים שלה.
    //                                 הנחה:  הרשימה מורכבת לפחות מאיבר אחד.

    //                                 הסבר הפתרון: בפתרון יש שימוש בשני מספרים הראשון
    //    הוא תחילת כל רצף(בי) והאחרון הוא סוף כל רצף(אף).לרצף הראשון
    //   נכניס את ערכה של החוליה הראשונה למשתנה הראשון.לאחר מכן נעבור
    //    בלולאה על הרשימה עד האיבר האחד לפני האחרון וכאשר ערך החוליה
    //     הבאה הוא לא המספר העוקב של ערך החוליה הנוכחית נכניס למשתנה
    //  השני את ערך החוליה הנוכחית. לאחר מכן, נכניס לרשימת הטווחים את
    // הרצף הנוכחי ונשנה את המשתנה הראשון להיות המספר הראשון ברצף הבא
    //  בסוף יש טיפול מיוחד בחוליה האחרונה אנו מכניסים את הרצף האחרון
    //        שמתחיל מהמשתנה הראשון וסופו הינו ערכו של החוליה האחרונה
    List<RangeNode> L = new List<RangeNode>();
    Node<int> pos1 = sourceList.GetFirst();
    Node<RangeNode> pos2 = null;
    int b = pos1.GetInfo(), f;
    while (pos1.GetNext() != null)
    {
        if (pos1.GetInfo() + 1 != pos1.GetNext().GetInfo())
        {
            f = pos1.GetInfo();
            pos2 = L.Insert(pos2, new RangeNode(b, f));
            b = pos1.GetNext().GetInfo();
        }
        pos1 = pos1.GetNext();
    }
    L.Insert(pos2, new RangeNode(b, pos1.GetInfo()));
    return L;

}
```

שאלה 3:

פתרון בשפת C# - נכתב ע"י דיתה אוהב-ציון:

א.

```
class Message
{
        private static int CountMessage = 0;  // משתנה סטטי - מונה הודעות ( מיספור אוטומטי)
        private int Id;                        // מספר סידורי של ההודעה
        private string Name;                   // שם השולח
        private int Size;                      // גודל ההודעה
}
```

ב.

```
class MessageBox
{
        private string Owner;                       // שם בעל התיבה
        private const int MaxSize = 100;            // גודל מקסימלי
        private List<Message> Box;                  // רשימת הודעות פעילות
        private List<Message> Bin;                  // רשימת הודעות בסל האשפה
}
```

הנחות: גודל התיבה מוגדר כקבוע - זהה לכל התיבות.
בתיבת ההודעות הפעילות , ההודעות ממוינות בסדר יורד. ההודעה החדשה ביותר - ראשונה.
בסל האשפה ההודעות ממוינות בסדר יורד.

ג.

```
public bool AddMessage(Message m)
{
        int freeSise= MaxSize - (GetActiveSize() + GetBinSize());
        // אם יש מקום פנוי בתיבה
        if (freeSise >= m.GetSize())
        {
                Box.Insert(null, m);
                return true;
        }

        //אם המקום הפנוי+ גודל ההודעות בסל האשפה  גדול או שווה לגודל ההודעה החדשה
        if (GetBinSize()+ freeSize >= m.GetSize())
        {
                //מחיקת הודעות מסל האשפה עד שיתקבל המקום הפנוי הנדרש
                while (freeSise < m.GetSize())
                        freeSise += RemoveFromBin();
                Box.Insert(null, m);
                return true;
        }
        return false;
}
```

ד. (1) גודל התיבה הוגדר ע"י חבר מחלקה קבוע

```
private const int MaxSize = 100;
```

כל הגולשים מקבלים תיבה בגודל הקבוע. אין הבדל בין הגולשים ואין צורך להקצות לכל גולש משתנה שישמור את גודל התיבה שלו. מאחר ויש דרישה להקצות תיבות בגודל משתנה יש לשנות אותו לחבר מחלקה רגיל `private int MaxSize;`
כך שניתן יהיה לשנות את ערכו לפי הדרישה והוא יהיה שונה לכל מופע של המחלקה - לכל תיבה יהיה גודל שונה.

(השינוי בבנאי המחלקה שבו נוסיף שורה `this.MaxSize = 100;`,
לאיתחול גודל התיבה.)

(2) אין צורך לשנות את הפעולה מסעיף ג . השימוש בשם המשתנה אינו מתשנה בביצוע הפעולה כאשר הוא הופך מקבוע למשתנה מחלקה רגיל.

שאלה 4:

פתרון בשפת Java:

qu

←	4	3	1	8	9	←

א. (1) טבלת מעקב ל- Sod1

qu לא ריק?	x	משפט זימון sod1(qu,st)	qu ← ←	st ⇄
T	4	sod1 ([3, 1, 8, 9] , [])	[9, 8, 1, 3, 4]	[4, 3, 1, 8, 9]
T	3	sod1 ([1, 8, 9] , [])	[9, 8, 1, 3]	[3, 1, 8, 9]
T	1	sod1 ([8, 9] , [])	[9, 8, 1]	[1, 8, 9]
T	8	sod1 ([9] , [])	[9, 8]	[8, 9]
T	9	sod1 ([] , [])	[9]	[9]
F				

(2) טבלת מעקב ל- Sod2

qu

←	9	8	1	3	4	←

st

→ ←	4	3	1	8	9

qu לא ריק?	x	משפט זימון sod2(qu,st)	y	qu ← ←	st ⇄
T	9	sod2([8, 1, 3, 4],[4 , 3 , 1 , 8 , 9])	9	[8, 6, 2, 16, 18]	[ ]
T	8	sod2([1, 3, 4],[4 , 3 , 1 , 8 , 9])	8	[8, 6, 2, 16]	[9]
T	1	sod2([3, 4],[4 , 3 , 1 , 8 , 9])	1	[8, 6, 2]	[8, 9]
T	3	sod2([4],[4 , 3 , 1 , 8 , 9])	3	[8, 6]	[1, 8, 9]
T	4	sod2([],[4 , 3 , 1 , 8 , 9])	4	[8]	[3, 1, 8, 9]
F					

qu

qu אחרי ביצוע קטע הקוד:

←	8	6	2	16	18	←

ב. sod1 הופכת את איברי התור (מהסוף להתחלה), ומעתיקה את איברי התור למחסנית, כך שהאיבר שהיה בראש התור יהיה בתחתית המחסנית.

ג. קטע הקוד מכפיל את כל איברי התור פי 2.

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'

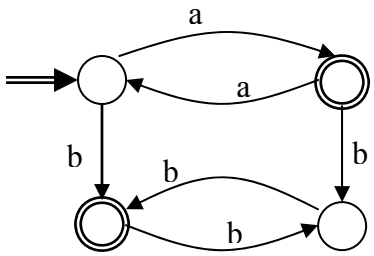
מודלים חישוביים

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

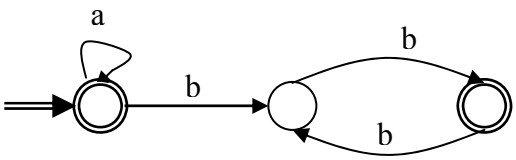
תרגיל 13:

- השפה L_1 אינה שפה רגולרית. יש תלות מניה בין אורכי הרצפים של a ו- b.

-השפה L_2 היא רגולרית, נבנה עבורה אוטומט סופי דטרמיניסטי:



- השפה L_3 היא שפה רגולרית , נבנה עבורה אוטומט סופי דטרמיניסטי:



ב. נתון ש-

$$L_4 = \{a^n b^{n-1} \mid n \geq 1,\ n\%2 = 1\}$$
$$L_5 = L_1 \cdot R(L_4)$$

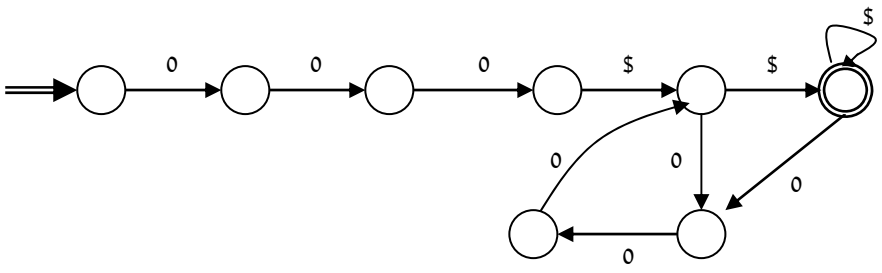
הגדרת השפה L_5:

$$\begin{aligned} L_5 &= \{a^n b^{n-1} \cdot b^{k-1} a^k \mid n,k \geq 1,\ k\%2 = 1\} \\ &= \{a^n b^{n+k-2} a^k \mid n,k \geq 1,\ k\%2 = 1\} \\ &= \{a^n b^t a^k \mid t = n+k-2,\ n,k \geq 1,\ k\%2 = 1\} \end{aligned}$$

שאלה 14:

א. המילה הקצרה ביותר היא $$000.

ב. האוטומט שמקבל את השפה L:



שאלה 15:

א.

1. (i) המילה 001001 מתקבלת.
מסלול חישוב:

$$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_4 \xrightarrow{0} q_5 \xrightarrow{0} q_6 \xrightarrow{1} q_7$$

(ii) המילה 01010 מתקבלת.
מסלול חישוב:

$$q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_4 \xrightarrow{0} q_5 \xrightarrow{1} q_6 \xrightarrow{0} q_7$$

(iii) המילה 0101 לא מתקבלת. (מילה מתקבלת באורך 4 חייבת להתחיל ב- 1)

2. האורך המינימאלי של מילה מתקבלת הוא 4. לדוגמא 1000.

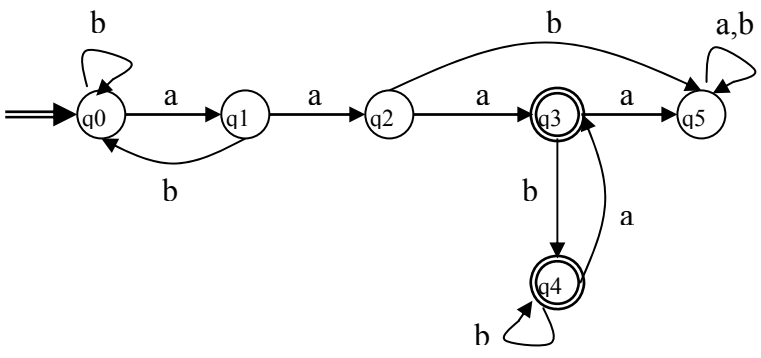
3. האורך המקסימאלי של מילה מתקבלת הוא 6. לדוגמא 011000.

4. השפה המוגדרת ע"י האוטומט היא:
כל המילים מעל $\{0,1\}$ שאורכן בין 4 ל- 6 (כולל) והתו הרביעי מהסוף הוא 1.
ניסוח אחר:

כל המילים המקיימות:

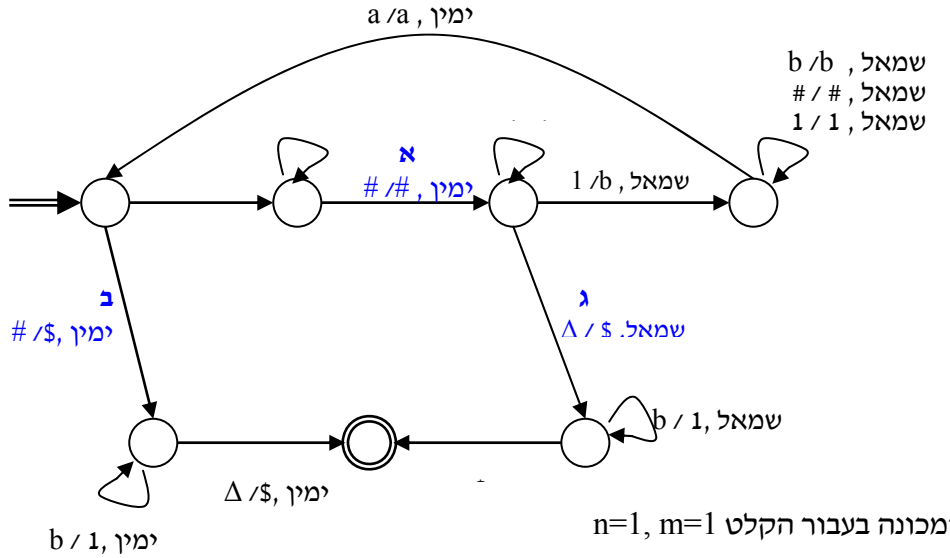
$$L = \{w_1 \cdot 1 \cdot w_2 \mid \quad w_1, w_2 \in \{0,1\}, 0 \le |w_1| \le 2, |w_2| = 3\}$$

ב. האוטומט המקבל את השפה L:



שאלה 16:

א. השלמת מכונת טיורינג עבור הפונקציה $f(m,n)=\min(m,n)$ כאשר $m,n>0$, כתובים בשפה האונרית וסימן # מפריד ביניהם. תוצאת המכונה תיכתב בין שני $.



ב. תהליך החישוב של המכונה בעבור הקלט n=1, m=1

⊢	1	#	1	Δ	

q_0

⊢	a	#	1	Δ	

q_1

⊢	a	#	1	Δ	

q_2

⊢	a	#	b	Δ	

q_3

⊢	a	#	b	Δ	

q_3

⊢	a	#	b	Δ	

q_0

⊢	a	$	b	Δ	

q_4

⊢	a	$	1	Δ	Δ

q_4

⊢	a	$	1	$	Δ

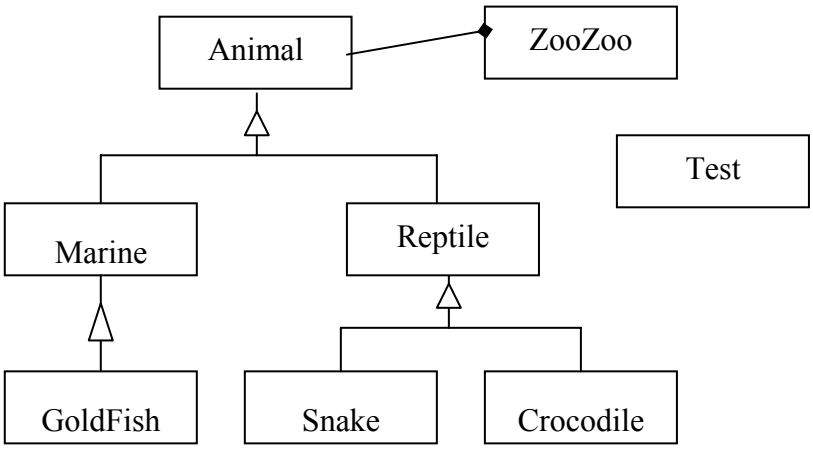
q_5

פרק ב'

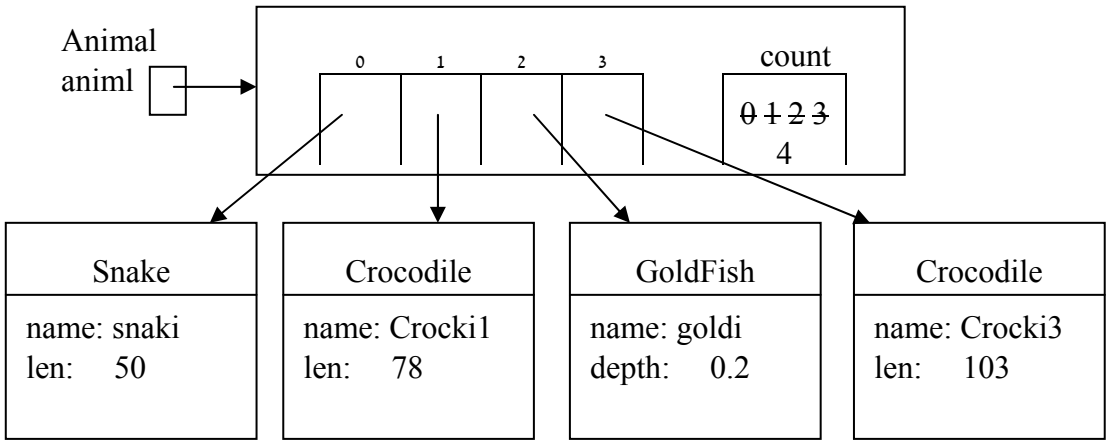
תכנות מונחה עצמים Java

תרגיל 17:

א.



ב.



פלט התכנית:

```
snaki is hungry !
Crawling: Yammi
Teeee
****
Crocki1 is hungry !
Crawling: Yammi
Finish eating Whaamm
****
goldi is hungry !
Swimming
Bloop bloop
****
Crocki3 is hungry !
Crawling: Yammi
Finish eating Whaamm
****
```

תרגיל 18:

א. כן.
WhatIn1 יורשת מ- WhatClass ולכן כל הפעולות של WhatClass הינן חלק מהפעלות של WhatIn1, ובכך היא עונה על התחייבותה לממש את הפעולה בממשק.
כלומר - WhatIn1 מתפקדת כ- WhatOp

ב. הפעולה לא תקינה. המחלקה יורשת מ- WhatClass ולכן עליה לזמן את הפעולה הבונה של מחלקת העל.
התיקון:

```
public WhatIn1(int number, int num)
{
      super(number);
      this.num = num;

}
```

ג. הפעולה לא תקינה. הרשאת הגישה לתכונה number היא private, ולכן היא מוסתרת מהמחלקה.
התיקון - זימון הפעולה getNumber():

```
public int calculate2()
{
      return (int)((this.getNumber() +this.num)/2);

}
```

ד. (1) אי אפשר להסתמך על הפעולה בונה ברירת מחדל.
מרגע שנוצרו פעולות בונות באחת ממחלקות-העל של WhatIn2, התבטלה האופציה של פעולה בונה ברירת מחדל. (אם נסיר את הפעולות הבונות של המחלקות WhatClass ו-WhatIn1 יופעל בנאי המחדל).

(2)
```
System.out.println(obj.calculate3(1000, 100, 10));
```

p1	p2	p3	calculate3(p1,p2,p3)		
1000	100	10	21,000 + 3* 10*10*10 24,000	calculate3(p1,p2)	
				1000 + 2*100*100 21,000	calculate3(p1)
					1*1000 1000

הפלט: 24,000

כל הפעולות מוכרות במחלקה WhatIn2 שיורשת מכל המחלקות האחרות.

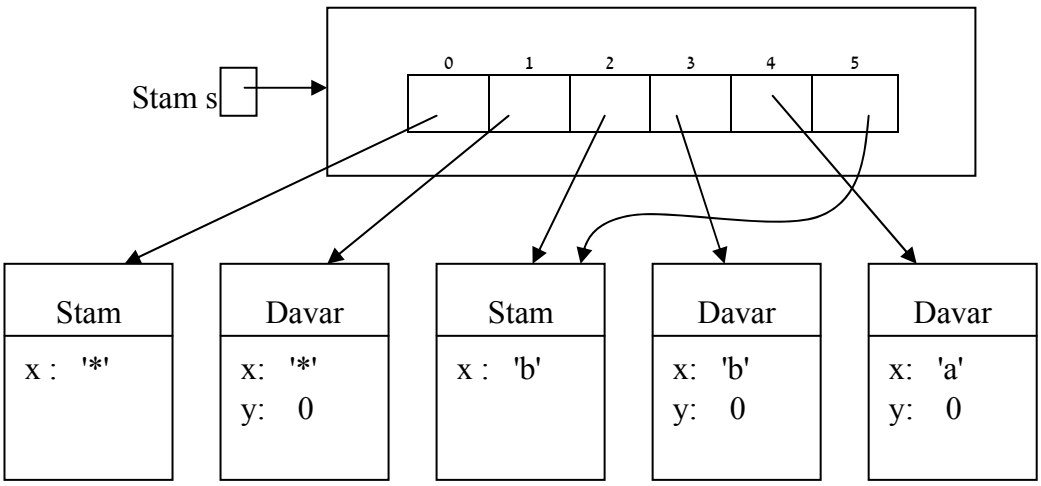
תרגיל 19:

א. (1) העיקרון של שתי פעולות בונות הוא **העמסה - overloading** .

(2) המהדר בוחר את הפעולה המתאימה לפי סוג ומספר הפרמטרים המועברים.

ב. הפעולה toString מגדירה מחדש (**דורסת - overriding**) את הפעולה שהועברה בירושה.
בתוך הפעולה יש זימון של הפעולה שהועברה **בירושה (inheritance)**.

ג. (1)



(2)

עמודה זו לא תודפס	פלט הלולאה בקטע 2:
s[0]:	x = *
s[1]:	Davar: x = *
s[2]:	x = b
s[3]:	Davar: x = b
s[4]:	Davar: x = a
s[5]:	x = b

(3) **פולימורפיזם** - מערך מסוג מחלקת העל מפנה לאובייקטים מסוג מחלקת העל ומסוג תת המחלקה (היורשת).
הורשה - אובייקט מתת-מחלקה מפעיל פעולה שירש ממחלקת העל (הפעולה ()print).

(4)

```
public boolean isStam1 (Stam other)
{
      return this.x == other.x ;
}

public boolean isStam2 (Stam other)
{
      return this.equals(other);
}
}
```

שאלה 20:

א. קטע קוד לבניית עצם בשם kind1 מסוג CandleKind:
שם הדגם: Rose
צבעי הדגם: מערך colors1 המכיל את בצבעים: אדום, צהוב, ירוק
מס' דגם: 899205
כמות מרבית: 2010

```
String[]colors1 = {"red", "yellow","green"};
CandleKind kind1 = new CandleKind("Rose", 899205, colors1, 2010);
```

ב. כותרת ותכונות המחלקה Factory:

prodLine - מערך פסי הייצור
maxLine - מספר מרבי של ספי ייצור
current - מספר פסי הייצור הקיימים בפועל
(בכל התכנית מתקיים: current < maxLine)

```
public class Factory
{
     private CandleKind [] prodLine;
     public static int maxLine = 12;
     private int current;
```

הפעולה הבונה: (לא נדרש בבחינה)

```
     public Factory()
     {
          this.prodLine = new CandleKind[maxLine];
          this.current = 0;
     }
}
```

ג. פעולות שיתווספו למחלקה CandleKind:

```
//---      סעיף ג' 1 ---
//--- פעולה המחזירה את צבע הנר מדגם נתון שיתחילו בייצורו ---

public String startColorProduction() { … }


//   ---      סעיף ג' 2 ---
//---  פעולה המחזירה את כמות הנרות המרבית
//---       שניתן לייצר מדגם הנרות הנוכחי ---

public int posibileProductAmount() { … }
```

פעולה שתתווסף למחלקה Factory:

```
//   ---      סעיף ג' 3 ---
//---        פעולה המחזירה את הקוד של דגם הנר שניתן לייצר ---
//--- זהו הנר שנמצא בכמות הקטנה ביותר במלאי, וניתן לייצרו ---
//---                  אם לא ניתן לייצר את הנר, יוחזר 999 ---

public int getCodeMinimumAmounts() { … }
```

ד. קטע הקוד בפעולה main():

```
Factory fty = new Factory();

int code = fty.getCodeMinimumAmounts();                    // (1)
while (code != 999)
{
     CandleKind ck = fty.getProdLine(code);                // (2)
     String colorToProduce = ck.startColorProduction();    // (3)
     int amount = ck.posibileProductAmount();              // (4)
     ck.update(colorToProduce, amount);
     code = fty.getCodeMinimumAmounts();                   // (1)

}
```

פרק ב'

תכנות מונחה עצמים C#

הפתרון לפרק זה נכתב ע"י **טובי סטפ**

תרגיל 21:

תרגיל 22:

תרגיל 23:

שאלה 24: