

מדעי המחשב

פרק ראשון

שאלה 1

Java

```
//--- ט.כניסה: מערך של מספרים שלמים ---
//--- ט. יציאה: מוחזר מקומו של המקסימום במערך ---
public static int bigB (int [] arr)
{
    int max = arr[0];
    int place = 0;
    for (int i = 1; i < arr.length; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
            place = i;
        }
    }
    return place;
}
```

```
//--- ט.כניסה: מערך של מספרים שלמים ---
//--- ט. יציאה: מוחזר מקומו של המקסימום במערך ---
public static int bigA (int [] arr)
{
    int max = 0;
    for (int i = 1; i < arr.length; i++)
    {
        if (arr[i] > arr[max])
            max = i;
    }
    return max;
}
```

C#

```
//--- ט.כניסה: מערך של מספרים שלמים ---
//--- ט. יציאה: מוחזר מקומו של המקסימום במערך ---
public static int bigB(int[] arr)
{
    int max = arr[0];
    int place = 0;
    for (int i = 1; i < arr.Length; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
            place = i;
        }
    }
    return place;
}
```

```
//--- ט.כניסה: מערך של מספרים שלמים ---
//--- ט. יציאה: מוחזר מקומו של המקסימום במערך ---
public static int bigA(int[] arr)
{
    int max = 0;
    for (int i = 1; i < arr.Length; i++)
    {
        if (arr[i] > arr[max])
            max = i;
    }
    return max;
}
```

Java

```
public class Game
{
    private String gameName;    // שם המשחק
    private int numPlayers;     // מספר השחקנים
    private boolean isWater;    // האם במים?

    //--- בנאי ---
    public Game(String gameName, int numPlayers, boolean isWater)
    {
        this.gameName = gameName;
        this.numPlayers = numPlayers;
        this.isWater = isWater;
    }

    //--- פעולה המחזירה את שם המשחק ---
    public String getGameName()
    {
        return gameName;
    }
}

public class Country
{
    private String countryName; // שם המדינה

    //--- הגדרת אוסף המשחקים ---
    public static int size = 43; // מספר המשחקים
    private Game [] games;      // מערך המשחקים
    private int lastGame;       // מספר המשחקים בפועל

    //--- בנאי ---
    public Country(String name)
    {
        this.countryName = name;
        this.games = new Game[size];
        this.lastGame = 0;
    }

    //--- האם מדינה משתתפת בבחינה? ---
    public boolean isParticipate (String name)
    {
        int i = 0;
        while (i < this.lastGame)
        {
            if (this.games[i].getGameName().equals(name))
                return true;
            i++;
        }

        return false;
    }
}
```

C#

```
class Game
{
    private String gameName;    // שם המשחק
    private int numPlayers;     // מספר השחקנים
    private bool isWater;      // האם במים?

    //--- בנאי ---
    public Game(string gameName, int numPlayers, bool isWater)
    {
        this.gameName = gameName;
        this.numPlayers = numPlayers;
        this.isWater = isWater;
    }

    //--- פעולה המחזירה את שם המשחק ---
    public string GetGameName()
    {
        return gameName;
    }
}

class Country
{
    private String countryName;    // שם המדינה

    //--- הגדרת אוסף המשחקים ---
    public static int size = 43;    // מספר המשחקים
    private Game[] games;          // מערך המשחקים
    private int lastGame;          // מספר המשחקים בפועל

    //--- בנאי ---
    public Country(string name)
    {
        this.countryName = name;
        this.games = new Game[size];
        this.lastGame = 0;
    }

    //--- האם מדינה משתתפת בבחינה? ---
    public bool isParticipate(String name)
    {
        int i = 0;
        while (i < this.lastGame)
        {
            if (this.games[i].GetGameName().Equals(name))
                return true;
            i++;
        }

        return false;
    }
}
```

שאלה 3

ב. מטרת הפעולה: הכברה של ארתוסטנס (כברה = מסנת) חישוב והדפסת כל המספרים הראשוניים עד n.

(הפעולה מדפיסה גם את 1 למרות שאינו נחשב כמספר ראשוני)

```

2 : 0 1 2 3 0 5 0 7 0 9 0 11 0 13 0 15
3 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
4 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
5 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
6 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
7 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
8 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
9 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
10 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
11 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
12 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
13 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
14 : 0 1 2 3 0 5 0 7 0 0 0 11 0 13 0 0
~~~~~
1 2 3 5 7 11 13

```

פלט הפעולה

פרק שני

אלף 4

א. פעולות הוספה ו-האם-קיים ב- $O(n)$ פעולות הצג-מינימום ו- הוצא-מקסימום ב- $O(1)$

- ייצוג: רשימה דו-כיוונית ממוינת (בסדר עולה או יורד) (בגודל n)
- הפנייה לאיבר בקצה אחד (מינימום) ולאיבר בקצה האחר (מקסימום).

מעבר על הרשימה למציאת המקום המתאים לאיבר. (המעבר $O(n)$ נעשה בעזרת שתי הפניות $pos - 1$ ו- $prev$ והכנסה אחרי $prev - O(1)$) עדכון ההפניות למינימום ולמקסימום במידת הצורך.	הוספה בצורה ממוינת insert (x) Insert (x)	$O(n)$
מעבר על כל הרשימה עד למציאת האיבר או עד שמגיעים לאיבר ראשון הגדול (או הקטן, תלוי בסוג המיון) ממנו.	האם-קיים? exists(x) Exists (x)	$O(n)$
החזרת ה- value של ההפניה למינימום. הנחה: הרשימה לא ריקה.	הצג-מינימום showMin() ShowMin()	$O(1)$
גישה מידית לאיבר המקסימום והוצאתו מהרשימה תוך עדכון ההפניה להפנות למקסימום החדש (הקודם במיון בסדר עולה / העוקב במיון בסדר יורד). הנחה: הרשימה לא ריקה.	הוצא-מקסימום getMax() GetMax()	$O(1)$

- ייצוג: מערך חד-ממדי ממוין בסדר עולה (המכיל n איברים)
- שמירת אינדקס לאיבר ראשון (מינימום) ולאיבר אחרון (מקסימום).

מעבר על המערך למציאת המקום המתאים לאיבר, הזזת האיברים כדי לפנות מקום לאיבר והכנסתו למערך. עדכון מקום המינימום והמקסימום במידת הצורך. הנחה: יש מספיק מקום במערך.	הוספה בצורה ממוינת insert (x) Insert (x)	$O(n)$
מעבר על כל המערך עד למציאת האיבר או עד שמגיעים לאיבר ראשון הגדול ממנו.	האם קיים? exists(x) Exists (x)	$O(n)$
החזרת הערך שאינדקס התא שלו נמצא במינימום. הנחה: המערך לא ריק.	הצג מינימום showMin() ShowMin()	$O(1)$
הוצאת האיבר האחרון במערך ועדכון ההפניה למקסימום (האיבר האחרון החדש) הנחה: המערך לא ריק.	הוצא מקסימום getMax() GetMax()	$O(1)$

ב. ביצוע פעולות הוספה ו-הוצא-מקסימום ב- $O(n)$ הפעולה $div7$ בסיבוכיות $O(1)$

- ייצוג: רשימה/שרשרת חד או דו-כיוונית, ממוינת (בגודל n)
- מונה למספר האיברים המתחלקים ב-7

$O(n)$	הוספה בצורה ממוינת insert(x) Insert(x)	מעבר על הרשימה למציאת המקום המתאים לאיבר. (המעבר $O(n)$ נעשה בעזרת שתי הפניות $prev$ ו- pos והכנסה אחרי $prev$ - $O(1)$) אם המספר מתחלק ב-7 - עדכון המונה.
$O(n)$	הוצא-מקסימום getMax() GetMax()	מעבר על כל הרשימה למציאת המקסימום והוצאתו. (ברשימה חד-כיוונית, המעבר נעשה בעזרת שתי הפניות $prev$ ו- pos) אם המספר מתחלק ב-7 - עדכון המונה. הנחה: הרשימה לא ריקה.
$O(1)$	מתחלק ב-7? div7() / Div7()	החזרת מצב המונה (החזר: מונה == 0)

- ייצוג: מערך חד-ממדי ממוין בסדר יורד
- מונה למספר האיברים המתחלקים ב-7 (המכיל n איברים)

$O(n)$	הוספה בצורה ממוינת insert(x) Insert(x)	מעבר על המערך למציאת המקום המתאים לאיבר, הזזת האיברים כדי לפנות מקום לאיבר והכנסתו למערך. אם המספר מתחלק ב-7 - עדכון המונה. הנחה: יש מספיק מקום המערך.
$O(n)$	הוצא-מקסימום getMax() GetMax()	המקסימום הוא האיבר הראשון, מחיקתו על ידי ציפוף שאר איברי המערך מקום אחד שמאלה (סגירת "החור"). אם המספר מתחלק ב-7 עדכון המונה. הנחה: המערך לא ריק.
$O(1)$	מתחלק ב-7 div7() / Div7()	החזרת מצב המונה (החזר: מונה == 0)

- ייצוג: מחסנית / תור ממוינים בסדר עולה
- מונה למספר האיברים המתחלקים ב-7 (בגודל n)

$O(n)$	הוספה בצורה ממוינת insert(x) Insert(x)	מעבר על איברי המחסנית/תור למציאת המקום המתאים לאיבר (כרוך בהעברת לכל היותר n איברים למחסנית/תור עזר והחזרתם). אם המספר מתחלק ב-7 עדכון המונה. הנחה עבור מחסנית/תור הממומשים במערך: יש מספיק מקום להוספת האיבר.
$O(1)$	הצג-מקסימום getMax() GetMax()	הערך המקסימלי נמצא בתחתית המחסנית / סוף התור. כרוך בהעברת כל n האיברים למחסנית / תור עזר והחזרתם. אם המספר מתחלק ב-7, עדכון המונה. הנחה: המחסנית / תור לא ריקים.
$O(1)$	מתחלק ב-7 div7() / Div7()	החזרת מצב המונה (החזר: מונה == 0)

שאלה 5

	0	1	2	3	4	5	length
c	2	1	1	1	0	2	6
	2	3	4	5	5	7	

* (1) א.

** (2)

	0	1	2	3	4	5
c	2	3	4	5	5	7
	1	2	3	4	4	6

	0	1	2	3	4	5	6	length
arr	5	0	2	1	3	0	5	7

	0	1	2	3	4	5	6	length
b	0	0	0	0	0	0	0	7
	0	0	1	2	3	5	5	

*** (3)

n	j	$j \geq n$	arr[j]	c[arr[j]] לפני	c[arr[j]] אחרי	$b[c[arr[j]] - 1]$
7	6	T	5	$c[5] = 7$	6	$b[7-1] = b[6] = 5$
	5	T	0	$c[0] = 2$	1	$b[2-1] = b[1] = 0$
	4	T	3	$c[3] = 5$	4	$b[5-1] = b[4] = 3$
	3	T	1	$c[1] = 3$	2	$b[3-1] = b[2] = 1$
	2	T	2	$c[2] = 4$	3	$b[4-1] = b[3] = 2$
	1	T	0	$c[0] = 1$	0	$b[1-1] = b[0] = 0$
	0	T	5	$c[5] = 6$	5	$b[6-1] = b[5] = 5$
	-1	F				

ב. הפעולה ממיינת את איברי מערך arr לתוך מערך b

ג. סיבוכיות הפעולה הוא $O(n)$:

הפעולה מבצעת 4 מעברים על 3 מערכים שכל אחד מהם בגודל n או k.

מעבר ראשון: אתחול מערך c ל-0 $O(k)$

מעבר שני *: מערך c משמש כמערך מונים

$O(n)$ לאיברי מערך arr. מניית האיברים

מעבר שלישי **: סכום האיברים במערך c $O(k)$

מעבר רביעי **: השמת כל ערך בתא המתאים במערך b $O(n)$

סה"כ $2n + 2k$ צעדים כך ש- $k \leq n$ ומכאן שמספר הצעדים חסום ע"י $4n$

↔ סיבוכיות ליניארית: $O(n)$

מיון מנייה או מיון ספירה (counting sort)

הוא אלגוריתם מיון עבור מספרים

שלמים המתבסס על העובדה שהמספרים

נמצאים בטווח חסום כדי לבצע את המיון בזמן

מהיר יותר מזה שמסוגלים לו אלגוריתמי המיון

הכלליים. בצורה אינטואיטיבית, די למיון לעבור

על קבוצת האיברים שרוצים למיין ולמנות את

מספר המופעים של כל אחד מהאיברים,

ומכאן שמו של האלגוריתם. (גוגל)

Java

```
//---          ? בעץ   ---
//--- פטולה המחזירה אמת אם x ---
//---          קיים בעץ ושקר אחרת ---
public static boolean exist (BinNode<Integer> bt, int x)
{
    if (bt == null)
        return false;
    if (bt.getValue() == x)
        return true;
    return exist (bt.getLeft(),x) || exist (bt.getRight(),x);
}

//--- פטולה המקבלת שני עצים בינאריים ורשימה ---
//--- ומכניסה לרשימה את כל האיברים שנמצאים ---
//--- בעץ הראשון ולא נמצאים בעץ השני ---
public static Node<Integer> check (BinNode<Integer> t1,
                                   BinNode<Integer> t2,
                                   Node<Integer> list)
{
    if (t1 != null)
    {
        int x = t1.getValue();
        if (!exist (t2, x))
            list.setNext (new Node<Integer> (x, list.getNext()));
        list = check (t1.getLeft(), t2, list);
        list = check (t1.getRight(), t2, list);
    }
    return list;
}
```

ניתן להחליף את קטע הקוד המסומן בקטע הבא

```
if (!exist(t2, x))
{
    Node<Integer> pos = list;
    pos.setNext(new Node<Integer>(x, pos.getNext()));
}
```

C#


```
// --- קיים בעץ ? ---
// --- פעולה המחזירה אמת אם x ---
// --- קיים בעץ ושקר אחרת ---
public static bool exist(BinNode<int> bt, int x)
{
    if (bt == null)
        return false;
    if (bt.GetValue() == x)
        return true;
    return exist(bt.GetLeft(), x) || exist(bt.GetRight(), x);
}

// --- פעולה המקבלת שני עצים בינארים ורשימה ---
// --- ומכניסה לרשימה את כל האיברים שנמצאים ---
// --- בעץ הראשון ולא נמצאים בעץ השני ---
public static Node<int> check (BinNode<int> t1,
                               BinNode<int> t2,
                               Node<int> list)
{
    if (t1 != null) {
        int x = t1.GetValue();
        if (!exist(t2, x))
            list.SetNext(new Node<int>(x, list.GetNext()));
        list = check(t1.GetLeft(), t2, list);
        list = check(t1.GetRight(), t2, list);
    }
    return list;
}

ניתן להחליף את קטע הקוד המסומן בקטע הבא
```

```
if (!exist(t2, x))
{
    Node<int> pos = list;
    pos.SetNext(new Node<int>(x, pos.GetNext()));
}
```

פרק פי"א

מערכות מחשב ואסמבלי

הפתרון לפרק זה נכתב ע"י: רונית (מרציאנו) גל-אור

עאלה ד

.א

```
MOV AX,7F80H
CMP AH,AL
JL SMALL
```

BIG:

```
MOV BL,AH
JMP SOF
```

SMALL:

```
MOV BL,AL
```

SOF:

טבלת מעקב (1)

	AX		BX		CF	OF	SF	ZF
	AH	AL	BH	BL				
MOV AX,7F80H	7Fh	80h						
CMP AH,AL					1	1	1	
MOV BL,AH				7Fh				

בסיום הקטע : BL = 7Fh

(2) הקטע משווה בין תוכן אוגר AH ואוגר AL בהתייחס לתוכן האוגרים כמספרים מכוונים ואת הערך הגדול שם באוגר BL

(3) אם נחליף את ההוראה
 JL SMALL בהוראה
 JB SMALL ערכו של BL ישתנה ויהיה : 80H

ב.

לא נכון! - הערכים שווים	10011110101110 2 >27AE16 (1)	
לא נכון!	27AE16 < 27AF16 <- 27AE16 > 1015910 (2)	
לא נכון! (ההסברים הפוכים)	התפקידים של SP ו BP BP - אחראי על ה"טיול" במחסנית -SP - מצביע על ראש המחסנית	
לא נכון!	בסיום הקריאה והביצוע של כל הוראה ערכו של IP גדל תמיד ב 1 - IP גדל בהתאם לגודל הפקודה האחרונה אותו ביצע	
נכון!	קטע 2 MOV DX,0000H MOV AX,0064H DIV AX	קטע 1 MOV DX,0000H MOV AX,0064H DIV AL
לא נכון!	קטע 2 MOV AL, 5BH MOV CL, 9 ROL AL, CL	קטע 1 MOV AL, 5BH MOV CL, 8 ROL AL, CL

```
ARR1 DW 2025H,1061H,1492H,5777H,1948H
ARR2 DW 1984H,1601H,2914H,9999H,8491H
K DW 5
```

```
XOR SI,SI ;INDEX OF ARRAY 1 AND 2
MOV CX,K ;LOOP COUNTER
MOV DX,0 ;COUNT POLINDROM WORDS
```

AGAIN:

```
PUSH CX ;KEEP LOOP COUNTER
MOV AX,ARR1[SI]
MOV CL,4
```

```
XCHG AH,AL
```

```
MOV BH,AH
SHR AH,CL
SHL BH,CL
ADD AH,BH
```

```
MOV BL,AL
SHR AL,CL
SHL BL,CL
ADD AL,BL
```

```
CMP AX,ARR2[SI]
JNE CONT
INC DX
```

CONT:

```
INC SI
INC SI
POP CX
LOOP AGAIN
```

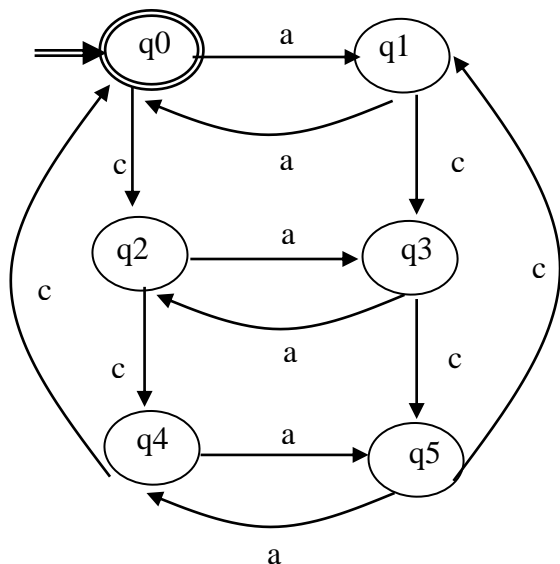
מודלים חישוביים

הפתרון לפרק זה נכתב ע"י: רחל לודמר / חיים אברבון

עבודה 11

א. ע"י רחל לודמר:

ניעזר במקרא הבא:

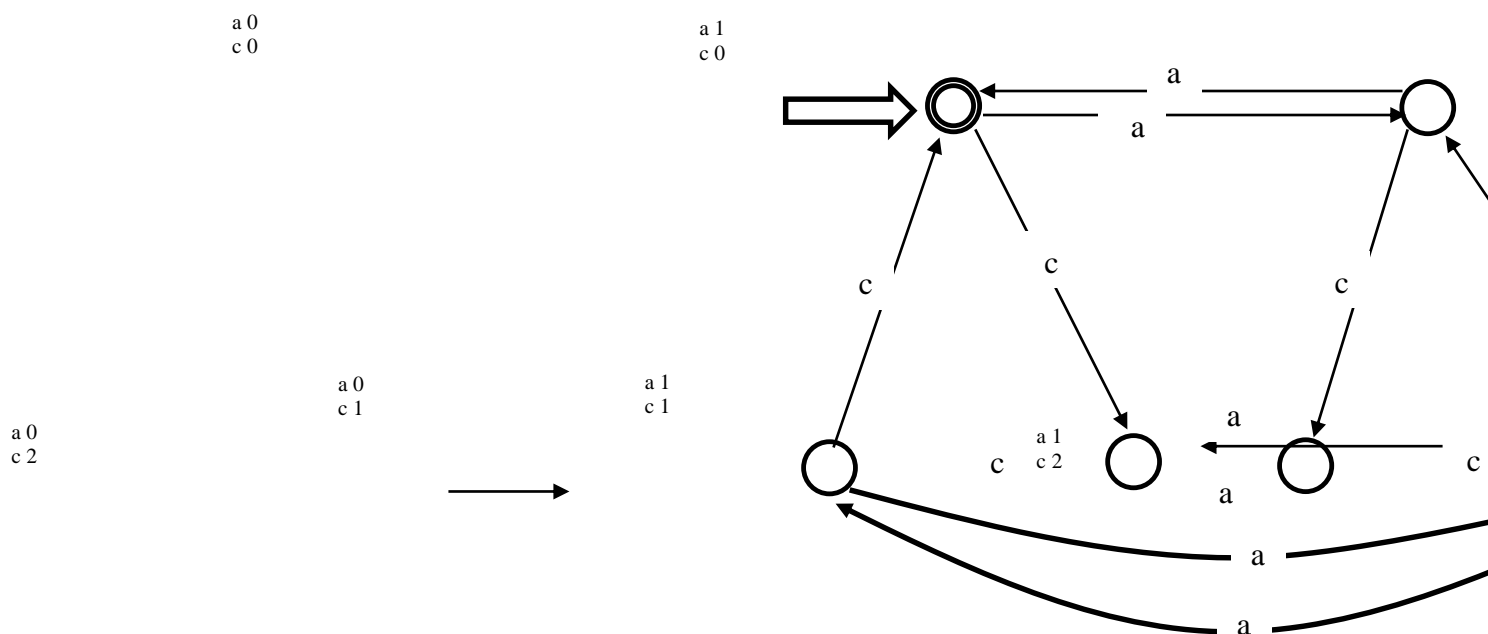


מצב	תאור: השאריות המתאימות של a, c עד למצב הנוכחי
q0	$ c \% 3 = 0, a \% 2 = 0$
q1	$ c \% 3 = 0, a \% 2 = 1$
q2	$ c \% 3 = 1, a \% 2 = 0$
q3	$ c \% 3 = 1, a \% 2 = 1$
q4	$ c \% 3 = 2, a \% 2 = 0$
q5	$ c \% 3 = 2, a \% 2 = 1$

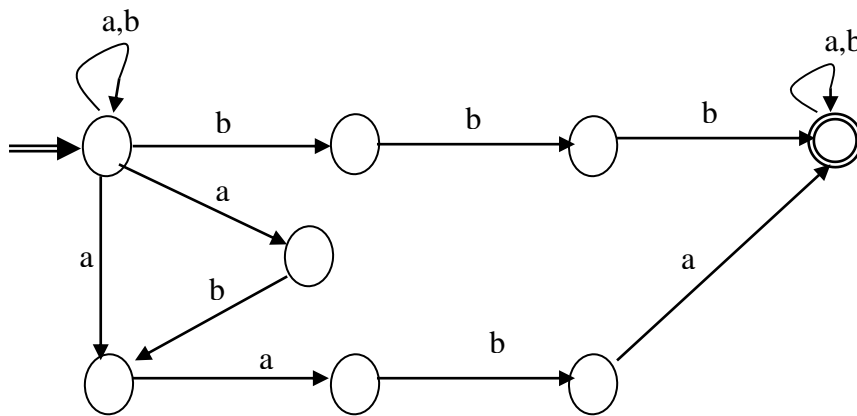
א. ע"י חיים אברבון:

(1) האוטומט מקבל את כל המילים מעל $\{a, c\}$ שמספר ה a ים זוגי ומספר ה c שארית חלוקה ב 3 שווה ל 0

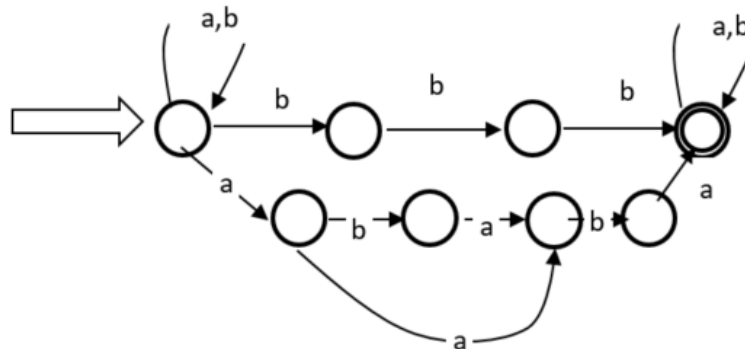
(2) כיון ששארית חלוקה ב 2 דורשת 2 מצבים ושארית חלוקה ב 3 דורשת 3 מצבים באוטומט המבוקש יהיו 6 מצבים. (ליד כל מצב רשום שארית החלוקה המתאימה).



ב. ע"י רחל לודמר :

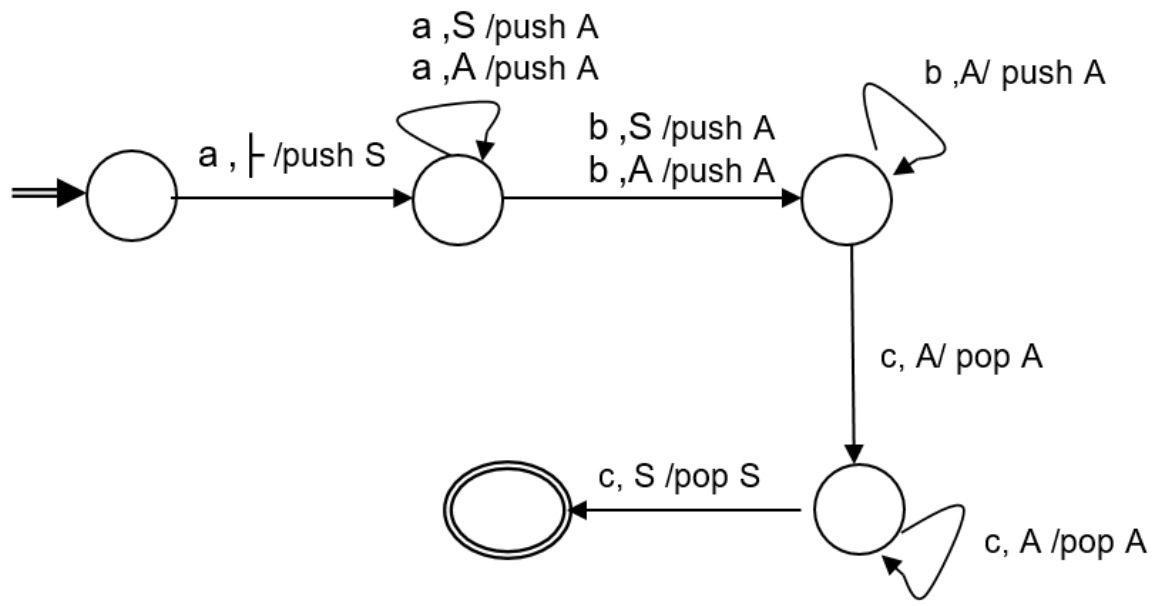


ב. ע"י חיים אברבונך :

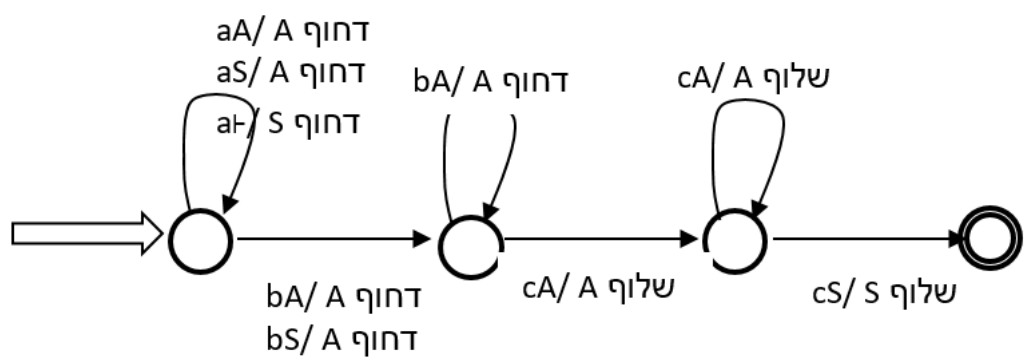


שאלה 12

א. ע"י רחל לודמר: אוטומט מחסנית

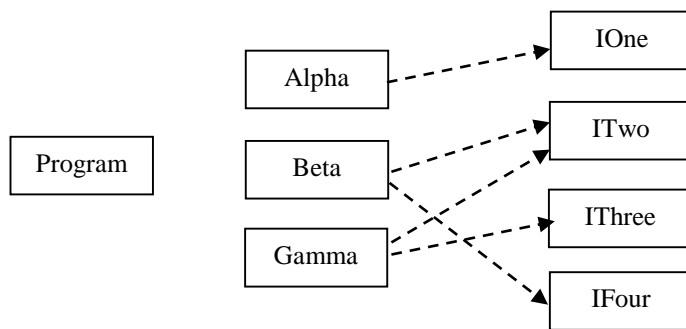


א. ע"י חיים אברבנד: אוטומט מחסנית



תכנות מונחה עצמים

15 / 13 אלה



א. כל מחלקה המממשת ממשק מתחייבת לממש את הפעולות שבממשק

מחלקה	פעולות Java	פעולות C#
Alpha	boolean firstA (Object x) void firstA (Object x)	bool FirstA (Object x) void FirstA (Object x)
Beta	int second() int fourth()	int Second() int Fourth()
Gamma	int second() int third()	int Second() int Third()

ב. ההגדרה אינה תקינה. יש לשמור על הסדר: מחלקה יורשת ורק אחר כך מממשת. התיקון:

```

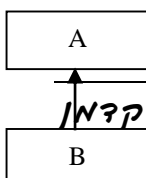
java: public class Omega extends Beta implements IFour { }
C#: public class Omega : Beta, IFour { }
    
```

ג.

i	ITwo x1 = new ITwo ();	שגוי. לא ניתן ליצור עצם מטיפוס ממשק
ii	Beta b = new Beta();	תקין. יצירת עצם מטיפוס המחלקה
iii	Alpha a = new Alpha(); IOne x2 = a;	תקין. Alpha מממשת את IOne. מתבצעת המרה כלפי מעלה מטיפוס המחלקה לטיפוס הממשק.
iv	Gamma c = new Gamma (); IOne x3 = c;	שגוי. Gamma אינה מממשת את IOne

ד.

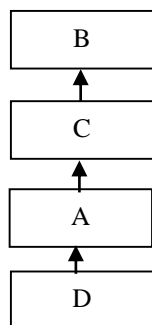
i	Beta b = new Beta(); ITwo b = new Beta(); int x = b.second ();	תקין. Beta מממשת את ITwo
ii	Gamma g = new Gamma(); Alpha a = (Alpha) g;	שגוי. אין קשר בין שתי המחלקות



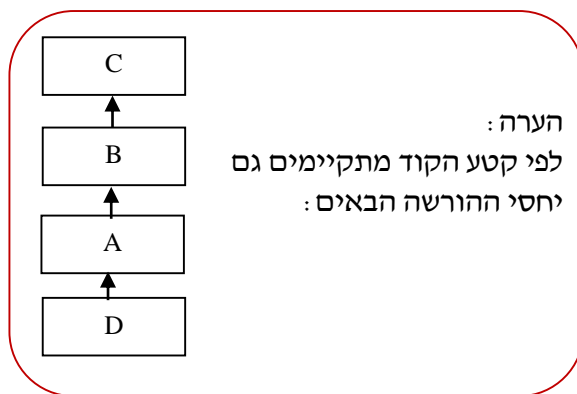
16 / 14 אלה

א. נתון: `A a1 = new B();`

- (i) קביעה לא נכונה. **המשפט תקין.** `Object obj = a1;`
 כל העצמים יורשים מ- `Object`. מתבצעת המרה אוטומטית כלפי מעלה.
- (ii) קביעה לא נכונה. **המשפט תקין.** `A a2 = a1;`
 ניתן לשים על אובייקט מטיפוס תת-מחלקה הפנייה מסוג מחלקת העל שלו.
- (iii) הקביעה נכונה. נגרשת המרה כלפי מטה: התיקון: `B b1 = (A) a1;`



ב. יחסי ההורשה הם: iv



ג. לכתוב רק `protected`. הערה: ניתן לכתוב גם `public`. זה יעבור קומפילציה אבל זה אינו תואם את כללי הסתרת המידע הנהוגים בתמ"ע.

ד. יש ליצור 5 עצמים, ולהפעיל את הפעולה על העצם השלישי, ואז יתקבל הפלט 3 ו- 5. ערך `num` של העצם השלישי הוא 3. התכונה המשותפת (הסטטית) `count` גדלה עם כל עצם חדש ולכן ערכה הוא 5.

הערה: הקוד של סעיף זה מטעה. למרות שהפעולה `printNow` מדפיסה שני ערכים, מכיוון שאין ביניהם רווח ניתן לקרוא את הפלט כמספר אחד: 35 (ולא כשני מספרים צמודים 3 ו- 5). חלק מהתלמידים הבינו שיש כאן שגיאה וחסר נתון ולכן לעולם לא יתקבל הפלט 35 (שלושים וחמש) וחלק מהתלמידים הבינו את השאלה כ-: "כמה עצמים יש ליצור על מנת שהמשתנה הסטטי יראה 35". הוספת רווח או פסיק בין המשתנים המוצגים במחרוזת הפלט היו פותרים את הבעיה.

בהצלחה!