

# מבנה הקורס

Monday, October 23, 2017

## תרגילי בית - 20%

תרגיל 0 - חובה

תרגילים 1-6 : לוקחים את ה-4 עם הציון הכי טוב.

## Quizzes - בחנים שבועיים - 2%

לוקחים לציון הסופי 3-ח הכי טובים מתוך ח (ח כנראה 14)

## בוחן אמצע - 15%

הבוחן חובה ב- 07/12 16:00-19:00

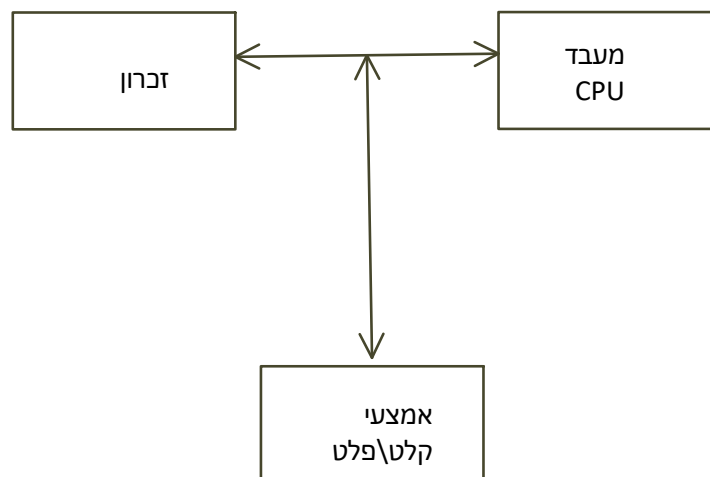
## מבחן סופי - 63%

**מחשב** - כלי אלקטרוני, הניתן לתכנות, שנועד לאחסן, לעבד ולאחזר את המידע.

**חומרה** - כל החלקים הפיזיים (מקלדת, מסך, ...).

**תוכנה** - כל מה שקשור לתכניות מחשב.

**מבנה המחשב** -



אלגוריתם הוא תהליך חישובי מוגדר היטב המיעד פתרון לבעיה מסוימת.

## קריטריונים לאלגוריתם

- סדרת פעולות לביצוע פתרון הבעיה.
- תיאור סופי.
- לא מתייחסים לנתוני קלט ספציפיים.
- קלט/פלט.

תכנות - תהליך בו מבטאים אלגוריתם בשפה שהמחשב יכול לבצע.

## דרכי תיאור האלגוריתם

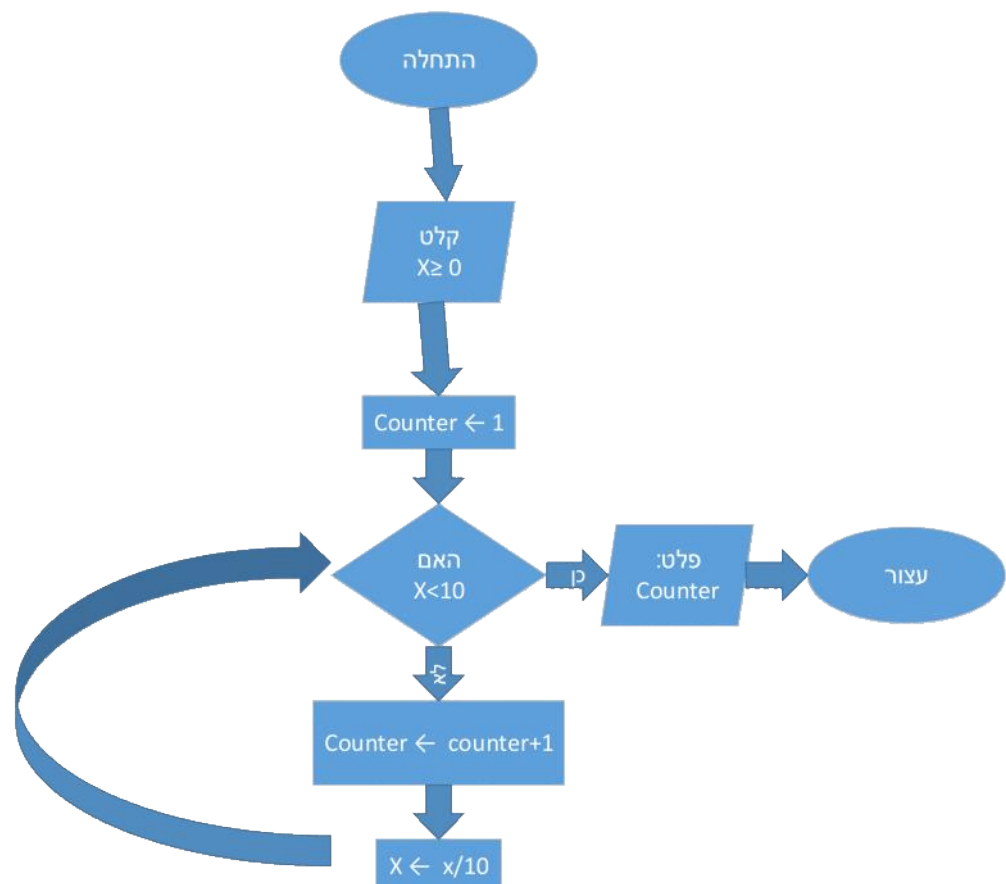
- במילים
- תרשים זרימה
- פסאודו-קוד : תיאור מופשט של אלגוריתם שדומה לקוד ומיועד לקריאה עבור בני אדם.

## דוגמא לתיאור אלגוריתם במילים

- בהנתן סדרה של מספרים  $a_1, a_2, \dots, a_k$  יש לחשב את הממוצע שלהם.
1. חבר את כל איברי הקלט ושמור את התוצאה במשתנה  $S$ .  $S = a_1 + a_2 + \dots + a_k$
  2. חלק את הסכום  $S$  במספר המחזורים  $K$ .  $x = S/K$
  3. עצור והחזר  $x$ .

## דוגמא לתיאור אלגוריתם בתרשים זרימה

בהנתן מספר שם חיובי, יש לחשב כמה ספרות יש בו.



**דוגמא לתיאור אלגוריתם בפסאודו-קוד**

בהנתן שני מספרים שלמים  $m$  ו- $n$  יש למצוא מחלק משותף מקסימאלי שלהם.

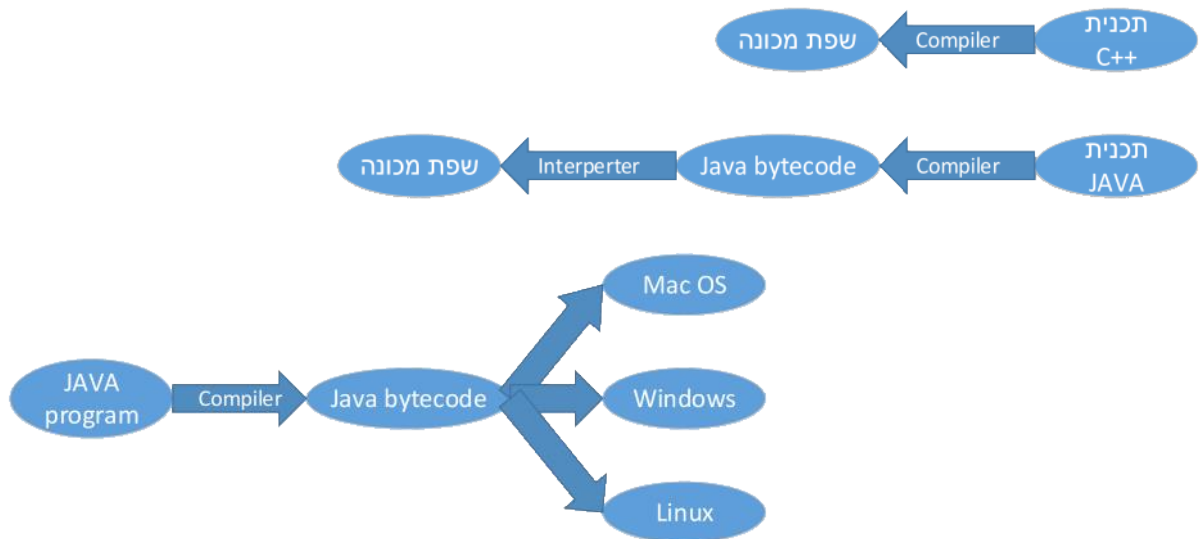
<u>אלגוריתם לא חכם</u>	<u>האלגוריתם של אוקלידוס</u>
1. קלט: $m > 0, n > 0$	1. קלט: $m > 0, n > 0$
2. $i \leftarrow m$	2. $r \leftarrow m \% n$
3. כל עוד לא מתקיים:	3. כל עוד $r \neq 0$
$i$ מחלק גם את $n$ וגם את $m$	3.1. $m \leftarrow n$
4. לולאה $i-1 < i$	3.2. $n \leftarrow r$
5. עצור. פלט: $i$	3.3. $r \leftarrow m \% n$
	4. עצור. פלט: $n$

**הוכחת תכונות של אלגוריתם אוקלידוס**

צ"ל:  $r = m \% n, \gcd(m, n) = \gcd(n, r)$

הוכחה: אם  $r$  היא שארית מחלוקה של  $m$  ב- $n$  אז ניתן לרשום  $m = x \cdot n + r$   
 נניח ש- $d$  הוא מחלק של  $m$  ו- $n$ . מכאן  $d$  הוא מחלק של  $r$ .  
 כלומר,  $d$  הוא מחלק משותף של  $n$  ו- $r$ .  
 כיוון שבי. נניח ש- $d$  הוא מחלק של  $n$  ו- $r$ . ונוכיח ש- $d$  הוא גם מחלק של  $r$  ו- $n$ .  
 כיוון ש-  $m = x \cdot n + r$ , אז גם  $m$  מתחלק ב- $d$ .  
 מש"ל

שפת מכונה: שפה שהמחשב מבין (...10011011001)  
 שפת עילית: שפה שמיועדת לבן אדם, בה לבן אדם נוח לתכנת ולהבין.  
Compile (מהדר): תכנית מיוחדת שמטרתה לתרגם משפת עילית לשפת המכונה או לשפה ברמה נמוכה יותר.



## תרגום פסאודו קוד ל-JAVA-

פסאודו קוד	שפת JAVA
Input: m,n>0   integer R -> m%n While (r!=0) M <- n N <- r R <- m%n Output: n	<pre> // This program calculates and prints the Gcd of two numbers import java.util.Scanner; public class Gcd {     public static void main(String[] args)     {         Scanner sc = new Scanner(System.in);         int m,n;         System.out.print("Enter first number:");         m = sc.nextInt();         System.out.print("Enter second number:");         n = sc.nextInt();         int r = m % n;         while (r != 0)         {             m = n;             n = r;             r = m % n;         }         System.out.println("The GCD is " + n);     } }                 </pre>

## תכנית שמוצאת את מספרת הספרות במספר שלם חיובי

```
import java.util.Scanner;
public class Digits {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int inputNumber;
        inputNumber = sc.nextInt();
        int counter = 0;
        while (inputNumber != 0) {
            inputNumber = inputNumber / 10;
            counter = counter + 1;
        }
        System.out.println("The given number has " + counter + " digits");
        sc.close();
    }
}
```

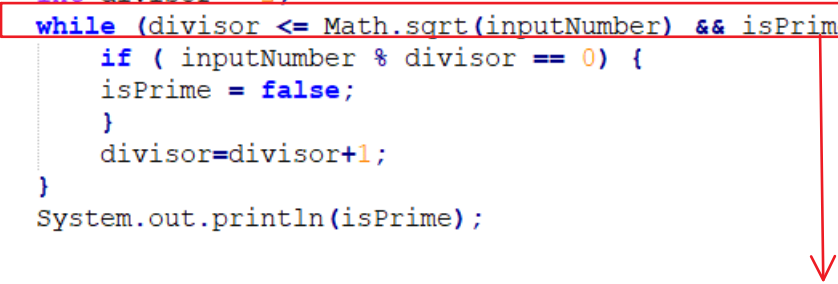
## תכנית שבהנתן מספר שלם חיובי גדול מ-1, בודקת אם הוא ראשוני

```
import java.util.Scanner;
public class IsPrime {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int inputNumber;
        System.out.print("Enter an integer number:");
        inputNumber = sc.nextInt();
        boolean isPrime = true;
        int divisor = 2;
        while (divisor <= Math.sqrt(inputNumber) && isPrime ){
            if ( inputNumber % divisor == 0) {
                isPrime = false;
            }
            divisor=divisor+1;
        }
        System.out.println(isPrime);
    }
}
```

## הדפסת מספרים ראשוניים (המשך)

תכנית שבודקת האם המספר ראשוני -

```
import java.util.Scanner;
public class IsPrime {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int inputNumber;
        System.out.print("Enter an integer number:");
        inputNumber = sc.nextInt();
        boolean isPrime = true;
        int divisor = 2;
        while (divisor <= Math.sqrt(inputNumber) && isPrime ){
            if ( inputNumber % divisor == 0) {
                isPrime = false;
            }
            divisor=divisor+1;
        }
        System.out.println(isPrime);
    }
}
```



✱ בלולאה רשמים `&&isPrime` כדי לעצור את התכנית אם המספר לא ראשוני.

✱ משתמשים בלולאה ב- `Math.sqrt(inputNumber)` במקום פשוט המסר, כדי לקצר את התהליך הפעולה ולא לבצע פעולות כפולות, כך שאם המספר הוא 48, יש לו מחלק- 3 ומכאן גם 16, וכדי למנוע בדיקה של שני המספרים אפשר לעשות בדיקה עד שורש המספר, שהוא בערך 6.

תכנית אשר מקבלת מספר שלם חיובי גדול מ-1 ומדפיסה את כל המספרים הראשוניים עד אליו -

```
import java.util.Scanner;
public class PrintPrimes {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an integer number:");
        int inputNumber = sc.nextInt();
        int nextNumber = 2; // first prime number
        // run from 2 to inputNumber
        while (nextNumber <= inputNumber) {
            boolean isPrime = true;
            int divisor = 2;
            while (divisor * divisor <= nextNumber && isPrime) {
                if ( nextNumber % divisor == 0) {
                    isPrime = false;
                }
                divisor = divisor + 1;
            }
            if(isPrime) {
                System.out.println(nextNumber);
            }
            nextNumber = nextNumber + 1; // move to the next number
        }
    }
}
```

משתנה (variable) - שם למקום בזכרון בו במהלך התכנית ניתן לשמור ערך/לקרוא משם ערך.

הכרזת המשתנה -	Type name; Int inputNumber = 8; boolean isPrime; Char firstLetter,secondLetter;
<ul style="list-style-type: none"> <li>יש להכריז כל משתנה, ניתן באותה שורה גם להכריז וגם לאתחל.</li> <li>שמות המשתנים חייבים להיות משמעותיים.</li> <li>אם שם המשתנה מורכב מכמה מילים, המילה הראשונה תתחיל באות קטנה וכל מילה אחריה תתחיל באות גדולה.</li> </ul>	

#### טיפוסים פרימיטיביים -

- נומריים:

שלמים:	Byte: מכיל בייט אחד
	Short: מכיל שני בייטים
	Int: מכיל ארבעה בייטים
	Long: מכיל שמונה בייטים

ממשיים:	Float: מכיל ארבע בייטים
	double: מכיל שמונה בייטים

- לא נומריים:

- Boolean - משתנה בוליאני יכול לקבל רק שני ערכים אפשריים: true,false.
- Char - תווים.

#### בקרת זרימה if-else -

```

If ( תנאי )
{
    הוראות שמתבצעות
    אם התנאי מתקיים
}
Else
{
    הוראות שמתבצעות
    אם התנאי לא מתקיים
}
    
```



## אופרטורים אריתמטיים

Operator	Use	Description
+	op1 + op2	Adds op1 and op2
-	op1 - op2	Subtracts op2 from op1
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes the remainder of dividing op1 by op2

## אופרטורים השוואתיים

Operator	Name	Description
x < y	Less than	true if x is less than y, otherwise false.
x > y	Greater than	true if x is greater than y, otherwise false.
x <= y	Less than or equal to	true if x is less than or equal to y, otherwise false.
x >= y	Greater than or equal to	true if x is greater than or equal to y, otherwise false.
x == y	Equal	true if x equals y, otherwise false.
x != y	Not Equal	true if x is not equal to y, otherwise false.

## המרת טיפוסים casting

מה קורה כשבביטוי מעורבים טיפוסים שונים?  
לפעמים JAVA מטפל בהמרה באופן אוטומטי, ולפעמים נרמה לבצע המרה יזומה.

המרה אוטומטית: byte -> short -> int -> long -> float -> double

דוגמא:  
Double a;  
Int b=7;  
a=b; //a=7.0  
b=a; ERROR

המרה יזומה:

Double a=3.5;  
Int b=7;  
a=b; //a=7.0  
b=(int) a; //b=3

## אופרטורים לוגיים

Operator	Name	Description
x && y	And	True if both x and y are true, otherwise false.
x    y	Or	True if at least one of x or y are true, otherwise false.
! x	Not	True if x is false, otherwise false.

## לולאת WHILE:

( תנאי שכל עוד הוא נכון הלולאה ממשיכה לרוץ ) While

```
{
    הוראות שמתצבעות בתוך הלולאה;
}
```

דוגמא לקוד שמתשמש בלולאת WHILE (התכנית מגרילה מספר שלם בין 1 ל-100 ומבקשת מהשתמש לנחש אותו)

```
import java.util.Scanner;
class Guess{
    public static void main (String[] args){
        Scanner sc = new Scanner(System.in);
        int solution, guess = 0;
        solution = 1 + (int) (Math.random() * 100);
        while ( guess != solution ){
            guess= sc.nextInt();
            if ( guess < solution )
                System.out.println ( "too small" );
            else if ( guess > solution )
                System.out.println ( "too big" );
        }
        System.out.println ( "You guessed it!" );
    }
}
```

## לולאת FOR

קודם כל מאתחלים משתנה והלולאה רצה כל עוד התנאי מתקיים עבור המשתנה המאוחל, כל סבב לולאה יש את תהליך הקידום שבדור"כ מגדיל את המשתנה ב-1.

For( קידום; תנאי; אתחול)

```
{
    הוראות שמתצבעות בתוך הלולאה;
```

דוגמא לקוד שמתשמש בלולאת FOR (במסיבה נפגשו 10 אנשים, כל אחד לחץ יד של כל אחד אחר, יש להדפיס את כל הלחיצות שנלחצו, ולחשב את מספרם)

```
class HandShakes{
    public static void main (String[] args){
        int i,j,counter=0;
        for (i=0;i<10;i=i+1){
            for (j=i+1;j<10;j=j+1){
                System.out.println(i+" handshaked "+j);
                counter=counter+1;
            }
        }
        System.out.println("There were "+counter+" handshakes");
        // n people => n*(n-1)/2 handshakes
    }
}
```

מערך הוא אוסף משתנים מאותו טיפוס המאוחזים תחת שם אחד. לכל משתנה כזה יש מספר סידורי. מספר סידורי במערך נראה index . ב-java אינדקסים הם מ-0 עד לגודל המערך פחות 1.

Int[] height = new int[10];

0	1	2	3	.	.	.	9
---	---	---	---	---	---	---	---



Height[2]=4

גישה לאיבר במערך -

Int[] score={87,89,45,80,67,57}

אתחול של איברי המערך בזמן ההכרזה:

87	89	45	80	67	57
----	----	----	----	----	----

ניתן לציין דמערך לא מצביע לאף כתובת על ידי מילה שמורה NULL: char[] letterGrades=null;

לכל מערך מוצמד קבוע בשם length המציין את גודל המערך int x=height.length; //x is 10

**דוגמא:** בקטע קוד הבא, משתמש מכניס גודל מערך, אחר כך איבר המערך. והתכנית מוצאת ומדפיסה על המסך אינדקס של האיבר המינימאלי במערך.

```
import java.util.Scanner;
public class minimalNumArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int size=sc.nextInt();
        int[] arr =new int[size];
        for(int i=0;i<size;i++)
            arr[i]=sc.nextInt();
        int minIndex=0;
        int minValue=arr[0];
        for(int i=1;i<size;i++)
        {
            if(arr[i]<minValue)
            {
                minValue=arr[i];
                minIndex=i;
            }
        }
        System.out.println(minIndex);
    }
}
```

**דוגמא:** הדפסת כל המספרים הראשוניים עד למספר שנקלט.

```
import java.util.Scanner;
public class PrintPrimesArray {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an integer number:");
        int inputNumber = sc.nextInt();
        int[] primes = new int[inputNumber]; // To keep the primes found so far
        int numberOfPrimes = 0; // Number of primes found so far
        for (int nextNumber=2; nextNumber<=inputNumber; nextNumber=nextNumber+1) {
            boolean isPrime = true; // Innocent till found guilty
            for(int primeIndex=0; primeIndex<numberOfPrimes && isPrime &&
                primes[primeIndex]*primes[primeIndex]<=nextNumber; primeIndex=primeIndex+1){
                if (nextNumber%primes[primeIndex] == 0) isPrime = false;
            }
            if (isPrime) {
                primes[numberOfPrimes] =nextNumber;
                numberOfPrimes = numberOfPrimes+1;
            }
        }
        for (int i=0; i<numberOfPrimes; i=i+1)
            System.out.println(primes[i]);
        sc.close();
    }
}
```

פונקציה מאפשרת למתכנת לחלק את התכנית לחלקים:

- כל חלק יהיה אחראי על ביצוע משימה מסוימת.
- כל חלק מקבל נתונים, מעבד אותם ומפיק תוצאה.
- יותר קל לתחזק בנפרד כל חלק מאשר לתחזק תכנית גדולה.
- מאפר חלוקת עבודה.
- מאפשר להשתמש בחלקי קוד שכתבנו בתכניות אחרות.

הכרזה על הפונקציה:

```
Public static <return type> <name> (<parameters>)
{
    פקודות;
}
```

בדיקת קלט - זריקת חריגה

אחראיות על קלט תקין מוטלת על מי שמשתמש בפונקציה, אם זאת לפעמים נרצה לעצור תכנית ולהדפיס שגיאה על המסך. ניתן לעצור תכנית בעזרת מנגנון שנקרא חריגה.

נכתוב בדיקה לתקינות הקלט ובמידה שהקלט לא תקין, נזרוק חריגה ונעצור את התכנית.

```
throw new RuntimeException("whatever");
```

**דוגמא:** תכנית למציאת מחלק משותף מקסימלי.

```
// This program calculates and prints the Gcd of two numbers
import java.util.Scanner;
public class Gcd
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int m,n;
        System.out.print("Enter first number:");
        m = sc.nextInt();
        System.out.print("Enter second number:");
        n = sc.nextInt();
        int result = gcd(m,n);
        System.out.println("The GCD is " + result);
    }
    // This function computes the greatest common divisor (gcd) of two positive numbers
    public static int gcd(int n, int m)
    {
        if(n <= 0 || m <= 0)
            throw new RuntimeException("one of the arguments is not positive");
        int r = m % n;
        while (r != 0)
        {
            m = n;
            n = r;
            r = m % n;
        }
        return (n);
    }
}
```

אם פונקציה לא מחזירה ערך, מציינים זאת על ידי מילה שמורה VOID.

**דוגמא:** פונקציה המקבלת מספר שלם חיובי ומחזירה מחלק שלו, או -1 אם אין מחלק כזה.

```
public static int findDivisor(int n){
    if(n <= 0)
        throw new RuntimeException("non-positive argument");
    int output = -1;
    for(int divisor = 2; divisor*divisor <= n & output == -1; divisor = divisor + 1){
        if(n%divisor == 0){
            output = divisor;
        }
    }
    return output;
}
```

**דוגמא:** פונקציה שמקבלת שני ארגומנטים n,divisor ובודקת אם n מתחלת ב divisor ללא שארית.

```
public static boolean isDivisor(int n, int divisor){
    if(n <= 0 | divisor <=1 | divisor>= n)
        throw new RuntimeException("illegal arguments");
    return n%divisor == 0;
}
```

**דוגמא:** פונקציה שמקבלת מערך ובודקת אם הוא ממוין

```
public static boolean isSorted(int[] array){
    if(array == null)
        throw new RuntimeException("input array is null");

    boolean sorted = true;
    for(int i = 0; i<array.length-1 & sorted; i = i + 1)
        if(array[i] > array[i+1])
            sorted = false;
    return sorted;
}
```

# העמסה של פונקציות - overloading

11:40 AM

Wednesday, November 15, 2017

ב-Java ניתן להגדיר פונקציות בעלי שם זהה, אך חייב להיות הבדל בפרמטרים שנשלחים לפונקציה.

מבנה הפונקציה:

<return type> <function name> (<parameters list>) → חתימת הפונקציה

חתימת הפונקציה חייבת להיות שונה בכל פונקציה.

דוגמא:

Public static int sum (int num1, int num2) { Return (num1+num2); }	Int res= sum(1,2);	Res = 3
Public static int sum (int num1, int num2) { Return (num1+num2); }	Int res=sum(1,2,3);	Res = 6

## חיפוש לינארי - linear search

נתון מערך וערך נוסף key, יש למצוא מהו הערך של key במערך ולהחזירו. במידה ו- key לא נמצא יש להחזיר -1.

לדוגמא:

המערך: 3,5,4,8,4,9,10

5 :Key

פלט: 1

דרך פתרון: נרוץ על כל איברי המערך וכל פעם נשווה את האיבר הנוכחי ל-key.

```
public static int linearSearch(int[] array, int key)
{
    if (array==null) throw new RuntimeException();
    int output = -1; // default (not found) value
    for (int i=0; i<array.length && output == -1; i=i+1)
    {
        if (key == array[i]) output = i;
    }
    return output;
}
```

## חיפוש בינארי - binary search

נתון מערך ממוין וערך key. יש להחזיר את האינדקס של key במערך, או -1 אם key לא נמצא.

לדוגמא:

המערך: 3,4,4,5,7,8,9,10,25

5 :Key

פלט: 3

דרך פתרון: הולכים לאמצע המערך ובודקים אם שווה ל-key אם לא בודקים אם גדול או קטן לו, לפי התשובה בוחרים חצי מערך שהאיבר נמצא בו, ומהתחלה.

```
public static int binarySearch(int[] arr, int key)
{
    if (arr==null) throw new RuntimeException();
    int output = -1; // default (not found) value
    boolean found = false;
    int start = 0, end = arr.length-1;
    while (start <= end && !found)
    {
        int middle = (start+end)/2;
        if(arr[middle] == key)
        {
            output = middle;
            found = true;
        }
        else if (key < arr[middle]) end = middle-1;
        else start = middle+1;
    }
    return output;
}
```

## מיון בחירה - selection sort

נתון מערך של ערכים. יש למיין אותו מהקטן לגדול.

לדוגמא:

מערך: 3,9,6,1,2

פלט: 1,2,3,6,9

דרך פתרון:

1. נמצא את האיבר בעל הערך המינימאלי במערך.
2. נחליף את העיבר שמצאנו עם האיבר הראשון במערך.
3. נמשיך באותה הצורה על שאר המערך, ללא האיבר הראשון.

נשתמש בפעולה swap בכדי להחליף בין שני איברים במערך

```
public static void swap(int [] arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

ובעט נריץ את הפעולה הממיינת

```
public static void selectionSort (int[] arr)
{
    if(arr == null) throw new NullPointerException();
    int minIndex;
    for (int i = 0; i < arr.length-1; i++)
    {
        minIndex = i;
        for (int j = i+1; j < arr.length; j++)
            if (arr[j] < arr[minIndex]) minIndex = j;
        swap(arr, minIndex, i);
    }
}
```



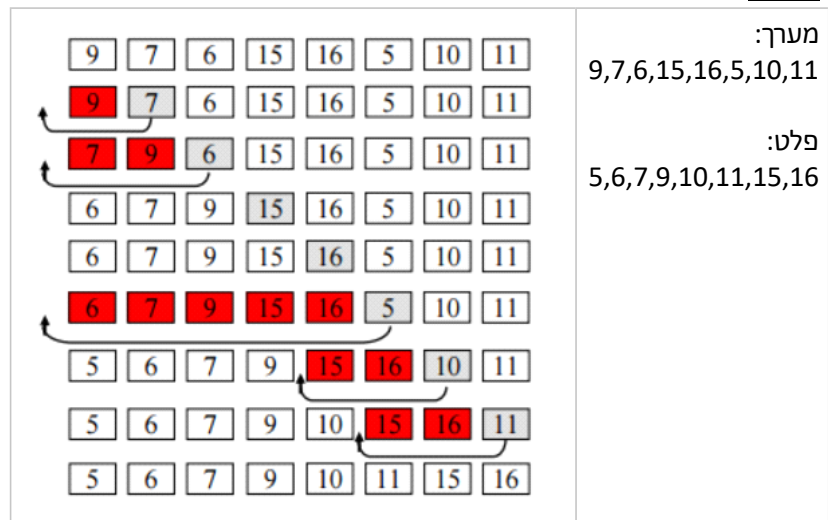
## מיון הכנסה - insertion sort

נתון מערך מספרים. יש למיין אותו מהקטן לגדול.

דרך פתרון:

1. בודקים אם האיבר השני גדול מהראשון, אם כן עוברים לאיבר הבא, אם לא מכניסים את האיבר הראשון למשתנה temp, שמים את הערך שבאיבר השני לתוך הראשון, ואת temp לאיבר השני.
2. עוברים לאיבר הבא ובודקים אם הוא קטן או גדול מזה לפניו, וחוזרים על שלב 1 בעצם עד שכל איבר מוצא את מקומו.

דוגמא:



פעולת מיון הכנסה:

```
public static void insertionSort (int[] arr)
{
    if(arr == null) throw new NullPointerException();
    for (int i = 1; i < arr.length; i++)
    {
        // insert arr[i] into a sorted sequence arr[0],..., arr[i-1]
        int key = arr[i];
        int j = i;
        // shift larger values to the right
        while (j > 0 && arr[j-1] > key)
        {
            arr[j] = arr[j-1];
            j--;
        }
        // insert key to its right j
        arr[j] = key;
    }
}
```

מחרוזת היא רצף של תווים הרשומים בין גרשיים כפולות ( " " ).  
דוגמאות למחרוזות: "I love Java", "Helo", " " ,

## הכרזה על מחרוזת

מחריזים על מחרוזת בצורה זו: `String str;`  
ההכרזה באה להגיד ש-`str` יכול הפנייה (כתובת) של מחרוזת בזיכרון. אך ההכרזה בניגוד למשתנים פרימיטיביים לא יוצרת מקום בזיכרון.

בדומה למערכים בכדי לתפוס מקום בזכרון למחזורת צריך להגדירה בצורה : `str= new String("Hello");`  
בנוסף ניתן באותה השורה גם להכריז על מחזורת וגם להגדירה: `String str2 = new String("Hello");`

- מחרוזות הן `immutable`, זאת אומרת שברגע שיצרנו והגדרנו מחרוזת, לא ניתן לשנות את התוכן.

## פונקציות של String

המחלקה `String` (בדומה למערכים) מגדירה מספר פונקציות שימושיות שאנו יכולים להתשמש בהם, לדוגמא:

`Length()` - פונקציה שמחזירה את אורך המחרוזת (מספר התווים).

<code>Int len = str.length();</code>	<code>len=5</code>
--------------------------------------	--------------------

`CharAt (int index)` - הפונקציה מקבלת אינדקס ומחזירה תו שנמצא באינדקס זה במחזורת.

<code>Char ch = str.charAt(0)</code>	<code>Ch='H'</code>
<code>Char ch = str.charAt(3)</code>	<code>Ch ="I"</code>

`Equals(string s)` - הפונקציה בודקת האם שתי מחרוזות זהות מבחינת תוכן.

<code>Str.equals(str2)</code>	<code>True</code>
<code>Str==str2</code>	<code>False</code>

(ניתן לחפש `Java API` בגוגל בכדי לראות עוד פונקציות שימושיות שמגיעות עם `JAVA`)

## הכרזה על מחרוזת ללא new

ניתן ליצור ולהגדיר מחזורת גם ללא שימוש באופרטור `new` בצורה הבאה:

```
String str3 = "Hello";
String str4 = "Hello";
```

ההבדל הוא שכאשר מגדירים מחזורת ללא `new` מתבצע `Java String Pool`, מה שאומר שכל פעם שיוצרים מחרוזת רק באמצעות `=`, המחזורת נשמרת במקום בזכרון הנקרא `Pool`, כך שבעצם לכל המחרוזות בתכנית שמגדירים בצורה זאת יש את אותה הכתובת בזכרון.

<code>Str3==str4;</code>	<code>True</code>
<code>Str3==str2</code>	<code>False</code>
<code>Str==str2</code>	<code>False</code>

## שרשור מחרוזות

פעולת שרשור מקבלת שתי מחרוזות ויוצרת מרוזת חדשה שהיא השרשות של שניהם, אך לא משנה את המחרוזות הישנות.

דוגמאות:

System.out.println("The answer is" + 7);	The answer is 7
System.out.println("The answer is" + (6+3));	The answer is 9
System.out.println("The answer is" + 6 + 3);	The answer is 63
System.out.println(6 + 3 + "is the answer");	9 is the answer

דוגמא: פונקציה שמקבלת מחרוזת str ותו ch וסופרת כמה פעמים ch מופיע בתוך str

```
public static int StringCounter (String str, char ch)
{
    int counter=0;
    for(int i=0;i<str.length();i++)
    {
        if(str.charAt(i)==ch) counter++;
    }
    return counter;
}
```

הגדרה רקורסיבית היא הגדרה בה מגדירים מושג באמצעים של עצמו.

A list is:

- A number - בסיס הרקורסיה
- Or
- A number, comma, list - הרקורסיה

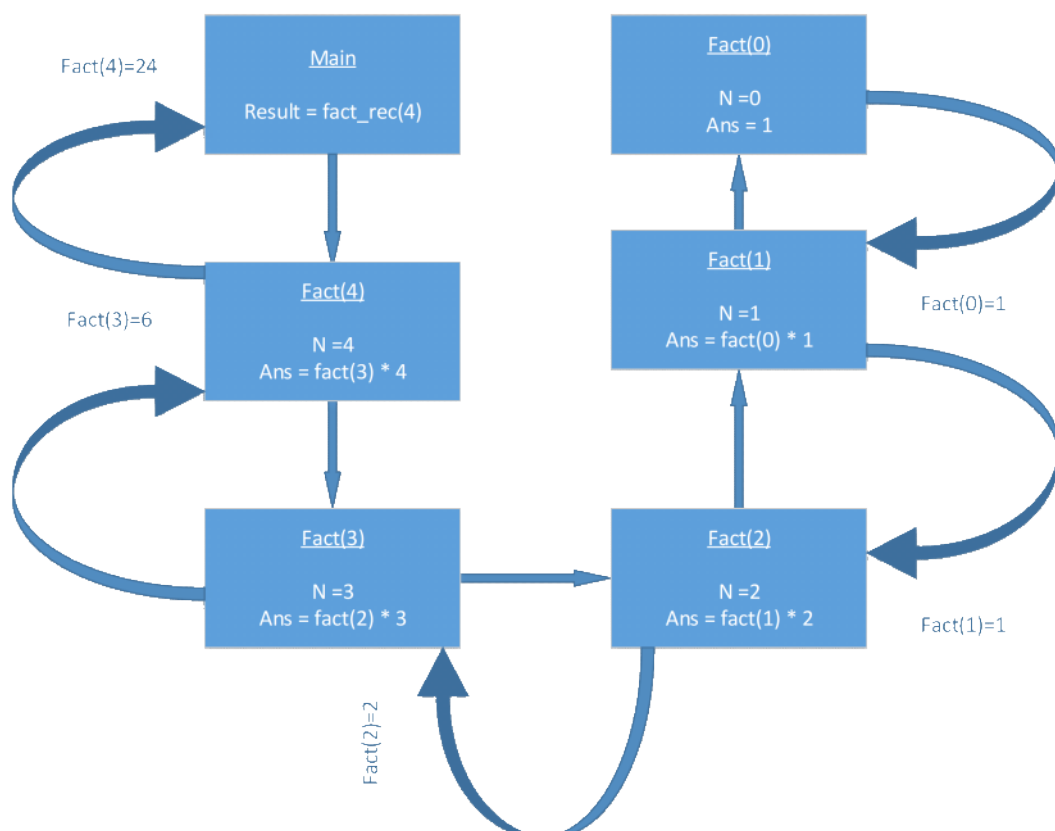
**דוגמא 1:**

רוצים לחשב את העצרת של  $n$  -  $n! = 1 * 2 * 3 * \dots * (n-1) * n = (n-1)! * n$

n!	1, n=0	- תנאי בסיס
	(n-1)! * n, n>0	- תנאי רקורסיבי

פונקציה לא רקורסיבית	פונקציה רקורסיבית
<pre>public static int fact(int n) {     int ans=1;     for(int i=2;i&lt;=n;i=i+1)         ans = ans*i;     return ans; }</pre>	<pre>public static int fact_rec(int n) {     int ans;     if (n==0)         ans = 1;     else         ans = n*fact_rec(n-1);     return ans; }</pre>

**מעקב אחר רקורסיה:**



## כללי אצבע לרקורסיות:

- (1) לסמוך על הקריאה הרקורסיבית
- (2) להתקדם לכיוון תנאי עצירה
- (3) לוודא שתמיד יש תנאי עצירה

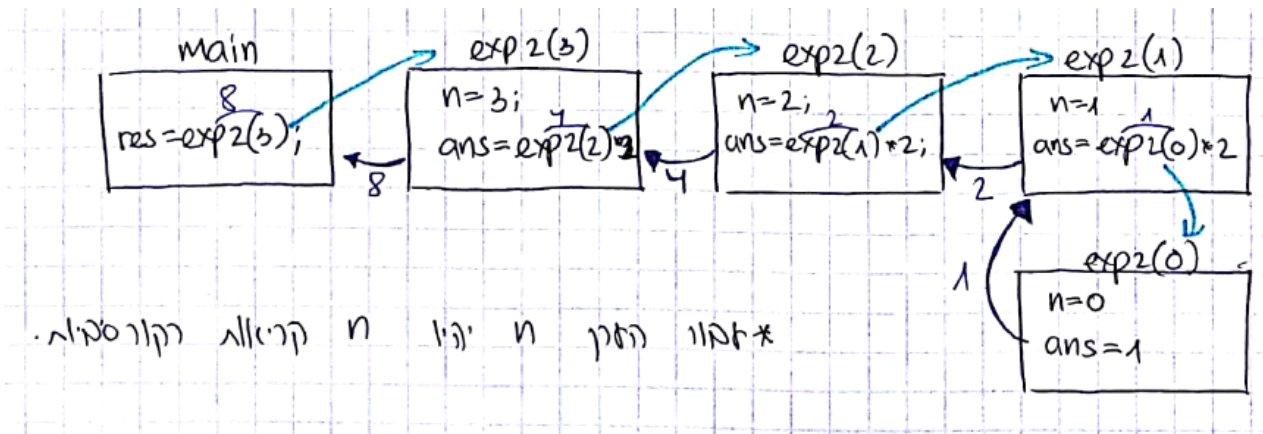
## דוגמא 2:

יש לכתוב פונקציה שמקבלת  $n > 0$  ומחזירה 2 בחזקת n.

```
public static int exp2(int n)
{
    int ans;
    if (n==0)
        ans = 1;
    else
        ans = 2*exp2(n-1);
    return ans;
}
```

$2^n$	תנאי בסיס - $1, n=0$
	תנאי רקורסיבי - $2^{(n-1)} * 2, n > 0$

## מעקב אחרי הפונקציה



## שיפור של דוגמא 2:

$$2^{16} = (2^8)^2$$

$$2^n = (2^{\frac{n}{2}})^2$$

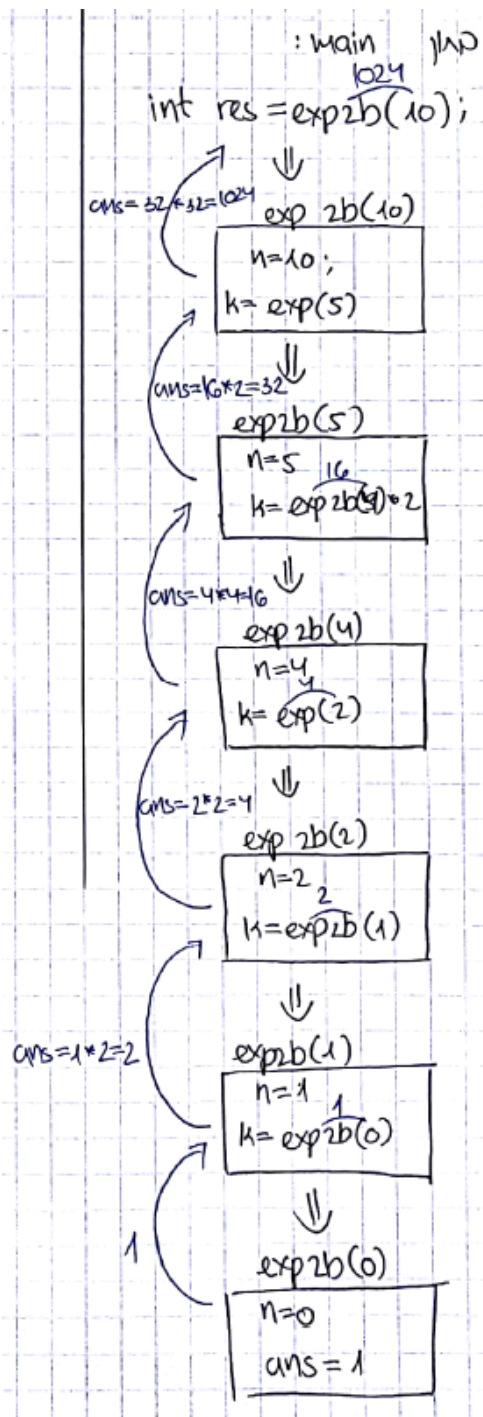
נשים לב שאם  $n$  זוגי, אז

$$2^n = \begin{cases} 1, & n=0 \\ (2^{\frac{n}{2}})^2, & \text{זוגי } n > 0 \\ 2^{n-1} \cdot 2, & \text{אז } n > 0 \end{cases}$$

## הפונקציה:

```
public static int exp2b(int n)
{
    int ans;
    if (n==0)
        ans = 1;
    else if (n%2 == 0) {
        int k = exp2b(n/2);
        ans = k*k;
    }
    else
        ans = 2*exp2b(n-1);
    return ans;
}
```

מעקב אחרי  
הפונקציה



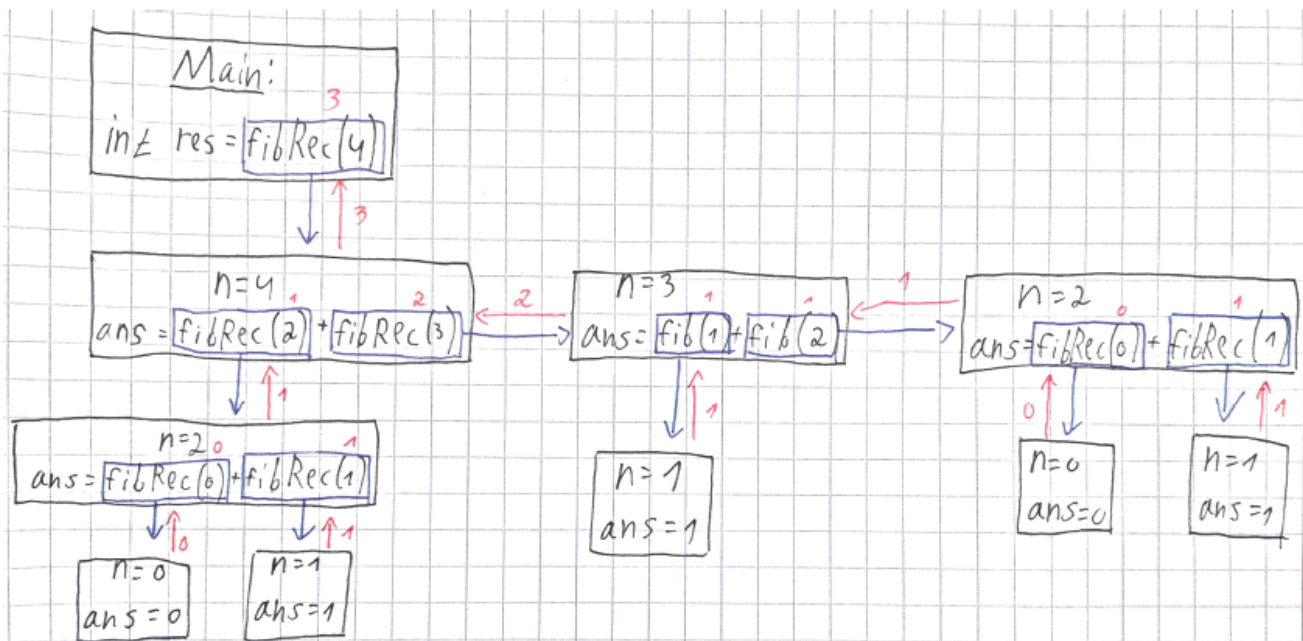
## סדרת פיבונצ'י

סדרת פיבונצ'י היא סדרה הפועלת על פי הכלל הבא:  $F_0=0, f_1=1, F_n = F_{n-1} + F_{n-2}, n>1$   
 הסדרה: 0,1,1,2,3,5,8,13,21,34,...

נרשום פונקציה שמקבלת  $n$  ומחשבת ומחזירה את המספר  $F_n$

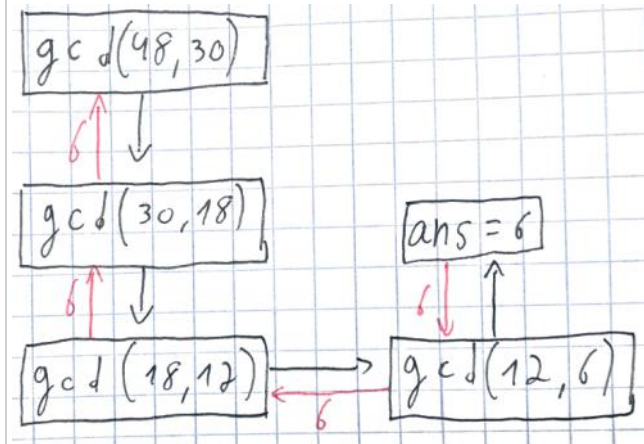
פונקציה איטרטיבית	פונקציה רקורסיבית
<pre> public static int fibIter(int n) {     int ans=0;     if(n==0)         ans=0;     else if(n==1)         ans=1;     else     {         int f0=0,f1=1;         for(int i=2;i&lt;=n;i=i+1)         {             ans = f0+f1;             f0=f1;             f1=ans;         }         return ans;     } }                 </pre>	<pre> public static int fibRec(int n) {     int ans;     if(n==0)         ans = 0;     else if(n==1)         ans=1;     else         ans = fibRec(n-1)+fibRec(n-2);     return ans; }                 </pre>

מעקב אחר הפונקציה



## דוגמא:

מעקב אחרי הפונקציה



נכתוב פונקציה שמוצאת מחלק משותף מקסימאלי (gcd) בעזרת רקורסיה

```

public static int gcdRec(int m, int n)
{
    int ans;
    if (m%n==0)
        ans = n;
    else
        ans = gcdRec(n, m%n);
    return ans;
}
  
```

## דוגמא:

נכתוב פונקציה המקבלת מערך array ואינדקס i ומחזירה אינדקס של האיבר המינימאלי בחלק array[i.....length-1]

```

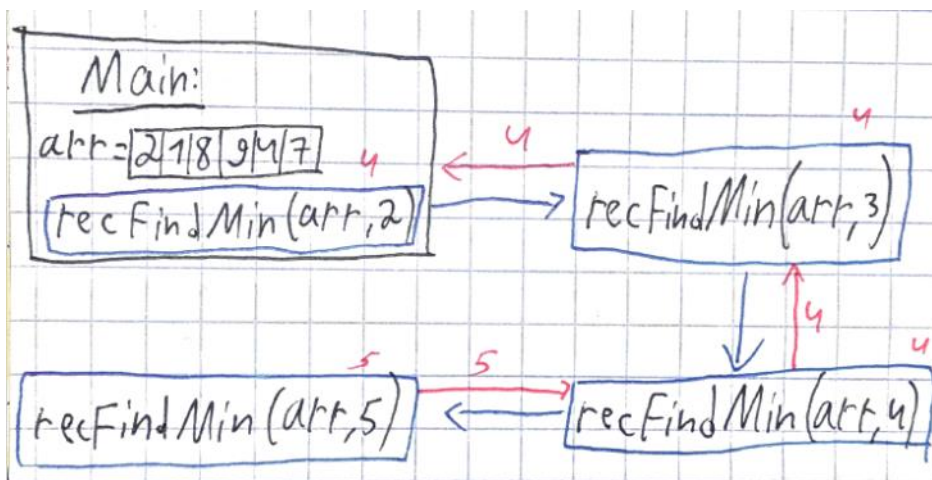
public static int recFindMin(int [] array, int i)
{
    int ans = i;
    if (i<array.length-1)
    {
        int j = recFindMin(array, i+1);
        if (array[j]<array[i])
            ans = j;
        else
            ans = i;
    }
    return ans;
}
  
```

רעיון:

אם  $i = \text{array.length} - 1$ , אז נחזיר את i (שזה חלק שמכיל איבר אחד בלבד)

אחרת נמצא מינימאלי בחלק array [i+1.....length-1]

ונשווה ערך זה עם איבר array[i]



מעקב אחר  
הפונקציה:



לפעמים במהלך כתיבת פונקציה רקורסיבית, מגלים שיש צורך להוסיף פרמטרים בקריאה לפונקציה. כדי להסתיר את הפרטים הטכניים, כתובים פונקציה שנקראת "פונקציית מעטפת" שהיא תהיה הממשק מול המשתמש, ופונקציית המעטפת תקרא לפונקציית עזר שתבצע את העבודה.

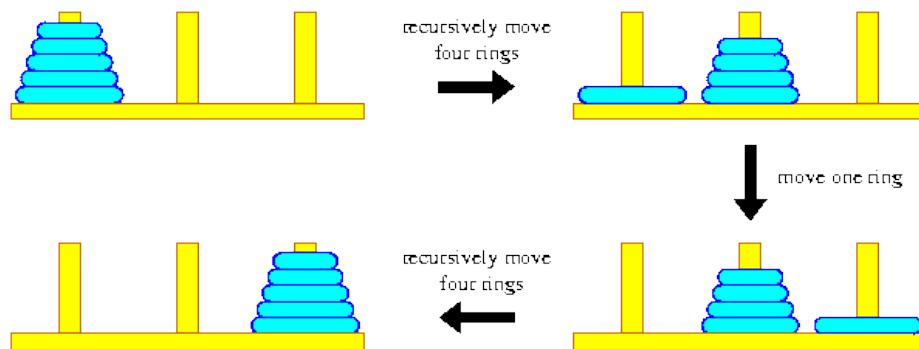
**דוגמא:** חיפוש בינארי

נתון מערך ממוין וערך key. יש להחזיר את האינדקס של key במערך, או -1 אם key לא נמצא.

<pre>public static int recSearch(int[] arr, int key) {     return recSearch(arr, key, 0, arr.length-1); }</pre>	פונקציית מעטפת
<pre>public static int recSearch(int[] a, int key,int start, int end) {     int ans = -1;     if (start &lt;= end)     {         int middle = (start + end)/2;         if (key &lt; a[middle])             ans = recSearch(a,key,start,middle-1);         else if (key &gt; a[middle])             ans = recSearch(a,key,middle+1,end);         else             ans = middle;     }     return ans; }</pre>	פינקציית עזר

# דוגמא לרקורסיה - מגדלי הנוי

9:37 AM Monday, November 27, 2017



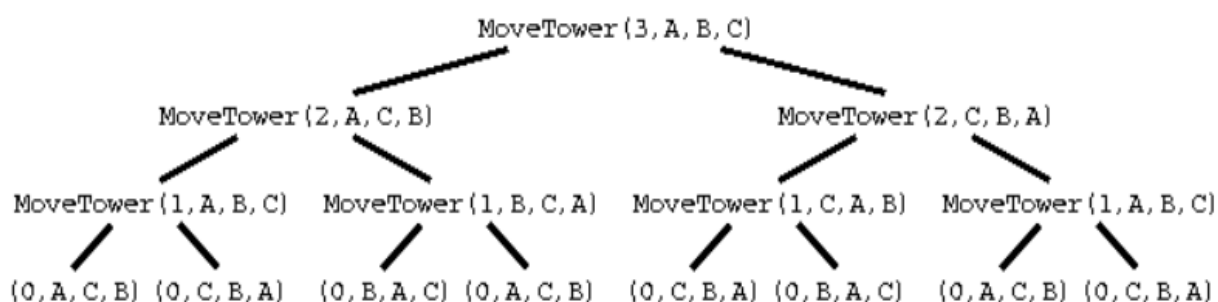
<p><b>מטרה:</b></p> <p>יש להעביר את הטבעות ממוט A למוט C בעזרת מוט B.</p>	<p><b>כללים:</b></p> <p>(1) אסור להניח טבעת על טבעת קטנה יותר.                  (2) בכל שלב ניתן להעביר טבעת אחת בלבד.                  (3) בכל שלב על הטבעות להיות על המוט, פרט לטבעת שמעבירים.</p>
---	--

<p><b>דרך פעולות עבור n כלשהו:</b></p> <p>- נעביר n-1 טבעות עליונות מ-A ל-B בעזרת C.                  - נעביר את הטבעת הגדולה ביותר מ-A ל-C.                  - נעביר n-1 טבעות מ-B ל-C בעזרת A.</p>	<p><b>דרך פעולה עבור n=3:</b></p> <p>- נעביר שתי טבעות עליונות מ-A ל-B בעזרת C.                  - טת הטבעת התחתונה נעביר מ-A ל-C.                  - נעביר שתי טבעות מ-B ל-C בעזרת A.</p>
--	--

## בתיבת הפונקציה:

```
public static void hanoi(int n, char source, char destination, char extra)
{
    if (n > 0)
    {
        hanoi(n-1, source, extra, destination);
        System.out.println("Move disk from "+source+ " to "+destination);
        hanoi(n-1, extra, destination, source);
    }
}
```

## מעקב אחרי הפונקציה:

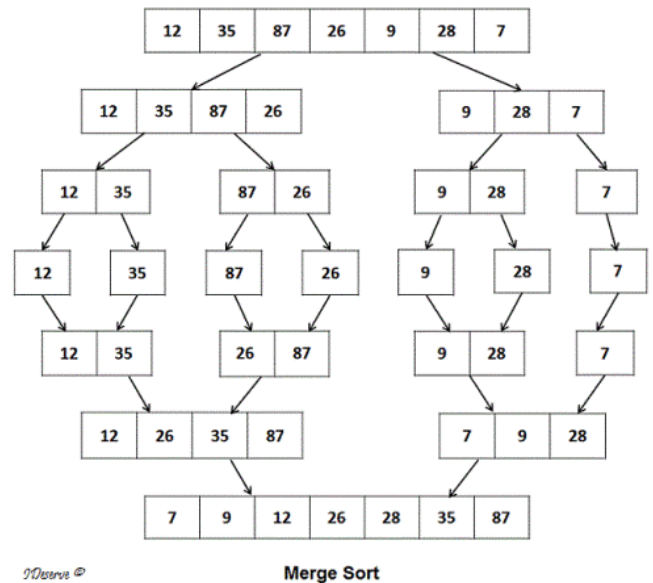


# דוגמא לרקורסיה - מיון מיזוג

9:37 AM

Monday, November 27, 2017

מטרה:	דרך עבודה:
למיון מערך.	(1) נחלק את המערך לשני חלקים.
	(2) נמיון את החלק השמאלי (נחזור על 1 ברקורסיה).
	(3) נמיון את החלק הימני (נחזור על 1 ברקורסיה).
	(4) נחבר ביחד את שני החלקים ונמיון ברקורסיה.



## רישום הפונקציה

```
public static void mergeSort(int[] arr, int low, int high)
{
    if(low<high) //more than one element
    {
        int middle= (low+high)/2;
        mergeSort(arr,low,middle);
        mergeSort(arr,middle+1,high);
        merge(arr,low,middle,high);
    }
}
```

## רישום פונקציית עזר שממזגת שני חלקים ממיונים במערך

```
public static void merge(int[] arr, int low, int middle, int high)
{
    int [] temp = new int[high-low+1];
    for(int i=0; low<=middle && (middle+1)<=high; i++)
    {
        if(arr[low]>arr[middle+1])
        {
            temp[i]=arr[middle+1];
            middle++;
        }
        else
        {
            temp[i]=arr[low];
            low++;
        }
    }
    for(i; row<=middle; i++) //elements left in arr[low...middle]
    {
        temp[i]=arr[low];
        low++;
    }
    for(i; middle+1<=high; i++)
    {
        temp[i]=arr[middle+1];
        middle++;
    }
    for(i=0; i<high-low+1; i++)
        arr[low+i]=temp[i]
}
```

# דוגמא לרקורסיה - מחליפה תווים במחרוזת

9:37 AM

Monday, November 27, 2017

## הוראות:

כתבו פונקציה רקורסיבית המקבלת מחרוזת str ושני תווים oldLetter, newLetter ומחליפה כל מופע של oldLetter ב- newLetter.

## דרך עבודה:

- יוצרים מחרוזת חדשה.
- בעזרת לולאה ורקורסיה משרשרים למחזורת החדשה newLetter על כל oldLetter שמוצאים במחרוזת הישנה, אם לא מוצאים משרשרים את אותו התו.
- הרקורסיה תעצור כשלא ישארו תווים במחרוזת הישנה.

## הפונקציה:

```
public static String replaceLetters(String str, char oldLetter, char newLetter) {  
    String ans;  
    if (str.length() == 0)  
        ans = "";  
    else {  
        char firstLetter = str.charAt(0);  
        if (str.charAt(0) == oldLetter)  
            ans = newLetter + replaceLetters(str.substring(1), oldLetter, newLetter);  
        else  
            ans = firstLetter + replaceLetters(str.substring(1), oldLetter, newLetter);  
    }  
    return ans;  
}
```

# דוגמא לרקורסיה - הדפסת תת מחרוזות

9:38 AM

Monday, November 27, 2017

## מטרה:

נכתוב פונקציה רקורסיבית שמדפיסה על המסך כל תת מחרוזת של המחרוזת הנתונה. (ניתן להניח שכל התווים שונים זה מזה).

## דוגמא:

s="abc"	תת מחרוזות: "ab", "ac", "bc", "a", "b", "ac"
	לא תת מחרוזת: "...ba", "cw"

## דרך פעולה:

- בונים תת מחרוזת.
- יש שתי אפשרויות, או לקחת את התו הראשון לתת מחרוזת שבונים.
- ניצור פעולת עזר שמקבלת מחרוזת ומחרוזת עזר.
- בפונקציה זו נשמור כל תתי המחרוזות שכן משרשרים למחרוזת העזר.
- בכל שלב מריצים שתי רקורסיות, אחת לוקחת את כל המחרוזת חוץ מהתא הראשון ולא משרשרת את התו הראשון. השנייה לוקחת את כל המחרוזת חוץ מהתא הראשון ומכניסה למחרוזת העזר את התו הראשון.

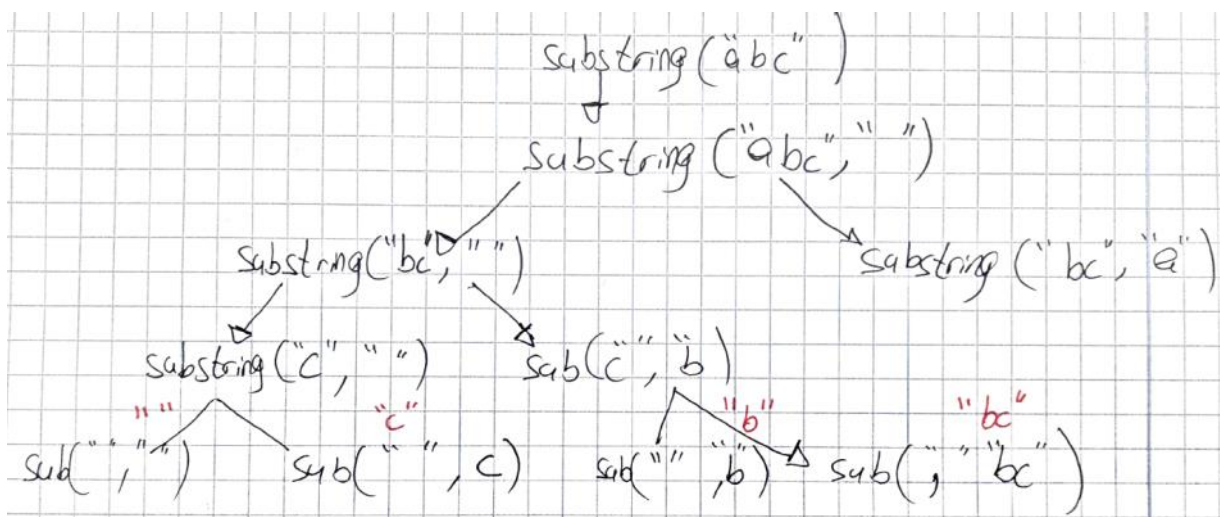
## פונקציית עזר:

```
public static void substrs(String s, String acc)
{
    if (s.length() == 0)
        System.out.println(acc);
    else
    {
        substrs(s.substring(1), acc + s.charAt(0));
        substrs(s.substring(1), acc);
    }
}
```

## פונקציית מעטפת:

```
public static void substrs(String s)
{
    substrs(s, "");
}
```

## מעקב אחרי הפונקציה:



# דוגמא לרקורסיה - הדפסת מסלולים באריג

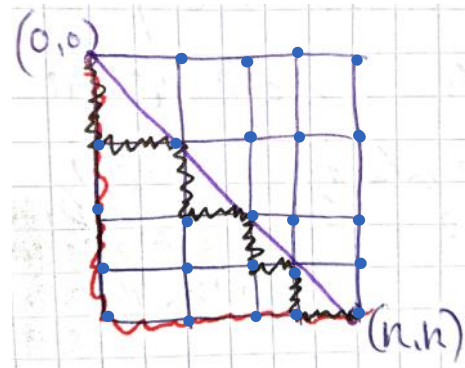
9:38 AM

Monday, November 27, 2017

מטרה:

- יש לכתוב פונקציה שמקבלת מספר שלם  $n$  ומחזירה את מספר המסלולים המונוטוניים האפשריים באריג  $n \times n$ .
- המסלול מתחיל בנקודה  $(0,0)$  ומסתיים בנקודה  $(n,n)$ .
  - המסלול בנוי מצעדים הנעים ימינה או למטה.
  - המסלול אינו עובר בנקודות מעל האלכסון.

דוגמא למסלולים נכונים:



דרך פעולה:

בכל נקודה שנמצאים בה, בודקים כמה מסלולים אפשריים יש למטה וימינה בעזרת שתי פעולות רקורסיביות, ומחברת בין כל התוצאות שיוחזרו.

פונקציית עזר:

```
public static int path(int n, int i, int j)
{
    int res = 0;

    if (i == n && j == n)
        res = 1;
    else if (i > n || j > n)
        res = 0;
    else if (i < j) // above the diagonal
        res = 0;
    else
    {
        res = path(n, i + 1, j) + path(n, i, j + 1);
    }

    return res;
}
```

פונקציית מעטפת:

```
public static int path(int n)
{
    return path(n, 0, 0);
}
```

# תכנות מובנה עצמים - OOP

11:15 AM

Wednesday, November 29, 2017

רוצים שתהיה לנו יכולת לאחד מספר דברים כטיפוס יחיד.  
לדוגמא: במקום לשלוח לפונקציה רשימה ארוכה של פרמטרים, נשלח פרמטר אחד הרבה נתונים.

כל פעם שמגדירים מחלקה חדשה, אנחנו בעצם מגדירים טיפוס חדש.  
למשתנים מטיפוסים המורכבים אלו קוראים אובייקטים/עצמים/מופעים של המחלקה (object).  
בהנתן אובייקט, נתייחס לשני חלקים מאוד חשובים:  
(1) מצב - המאפיינים של האובייקט.  
(2) התנהגות - פונקציות/שיטות שמשויכות לאובייקט.

לדוגמא:

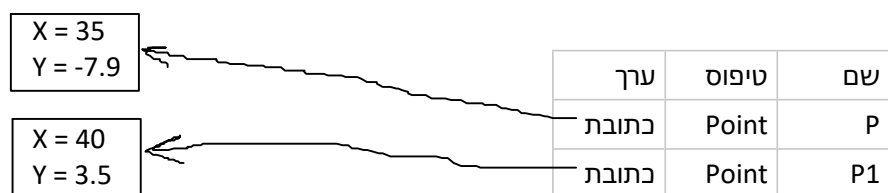
התנהגות	מצב	אובייקט
charAt(), subString()...	"abc"	String str1 = new String("abc")
charAt(), subString()...	"bcd"	String str1 = new String("bcd")
Length...	1,2,3	Int[] arr = {1,2,3}

## הגדרת טיפוס חדש

נניח שרוצים לעבוד עם נקודות במישור.  
נכריז על מחלקה חדשה בה נגדיר את המשתנים שיגדירו נקודה במישור.

Public class Point { public double x; public double y; }	המחלקה Point מגדירה טיפוס חדש עבור ייצוג נקודה. המשתנים x,y נקראים משתני המחלקה או שדות.
--	---

P1.x = 40; p2.y = 3.4;	p.x = 35; p.y = -7.9;	Point p = new Point(); Point p1 = new Point();	עכשיו ניתן להגדיר משתנים מטיפוס Point.
---------------------------	--------------------------	---	--



<pre>public static void printPoint(Point p) {     System.out.println("p="+p.x+" "+p.y); }</pre>	כעת ניתן לעביר את האובייקט כפרמטר לפונקציה
---	--