

## Contents

2	String
2	מחזיר את המספרים במערך למחרוזת
2	בודק אם צמד מילים הם אנגרם
2	temp Media
3	בודק אם יש מילה משותפת
3	מחזירה את המילה הארוכה וגדולה לקסגורפית במערך דו ממדי
3	//מחזיר את מספר ההופעות של המחרוזת המחרוזת השניה בראשונה
3	מזיז את כל ההופעות של איקס קטן לקצה המערך
4	מוצא את המילה שחזרת הכי הרבה פעמים במחרוזת
4	מחזיר את המחרוזת בלי תו שחוזר על עצמי
4	מחזיר את המילה הארוכה ביותר מבין המילים במחרוזת כאשר הם מופרדים ע"י שלוש כוכביות
4	סכום המספרים שמופיעים במחרוזת
5	מערבל את המילים
6	מחזיר את סכום הספרות במחרוזת
6	סופר את כמות ההופעות של שלוש תווים רצופים
6	Recursion:
6	מחזיר את ההפרש בין המספר הגדול לקטן במערך
6	זיג זג - לשים לב לא עובד
7	סכום המפרים המתחלקים בשלוש
7	floodones גרסא שלי
8	רצף הארוך ביותר של מספרים המתחלקים בשלוש שכולל מיקום מסויים במערך
8	מחזיר את מספר ההופעות של ספרה במספר
8	מחזיר את המחרוזת בסדר הפוך בלי נקודה
9	בודק אם לכל ספרה המספר הפוך לה נמצא
9	כל התמורות של אותיות גדולות וקטנות כך שסדר המילה נשמר
9	הדפסת שתי משלושים בעלי קודקוד משותף בסיס בגודל
9	מדפיס את המספר כך שהוא מתחיל מהמספר וכל פעם מוריד את ספרת האחדות
10	מחזיר את מספר הספרות התואמות
10	מדפיס מחרוזת כך שיופיעו כל התמורות האפשריות מהמחרוזת יש לשלוח את המחרוזת כמערך של תוים
10	משולש פסקל
11	מחזיר אם סכום חלק מהמערך שווה למטרה

- 11..... מחזיר אם סכום חלק מהמערך שווה למטרה מספק עוקב למספר שנבחר לא יכול להבחר
- 11..... כמות השלשות העוקבות במספר

## String

```

public static int[] numToArray(String s){
    מחזיר את המספרים במערך למחרוזת
    s=s.replaceAll("[!0-9]"," ");
    String[] A=s.split(" ");
    int[] B=new int[A.length];
    for(int i=0;i<B.length;i++){
        B[i]=Integer.parseInt(A[i]);
    }
    return B;
}

public static boolean Anagram(String s1,String s2){
    בודק אם צמד מילים הם אנגרם
    s1=s1.replaceAll(" ", "");
    s2=s2.replaceAll(" ", "");
    char[] A=s1.toCharArray();
    char[] B=s2.toCharArray();
    int i,j; boolean flag;
    for(i=0;i<A.length;i++){
        flag=false;
        for(j=0;j<B.length;j++){
            if(A[i]==B[j]){
                flag=true;
                B[j]=' ';
            }
        }
        if(!flag) return false;
    }
    return true;
}

public static int[] TempMedia(String s){
    temp Media
    String[] A=s.split(" ");
    int[] B=new int[A.length];
    B[0]=Integer.parseInt(A[0]);
    int i;
    for(i=1;i<A.length;i++){
        if (A[i].equals( "." )) B[i]=B[i-1];
        if (A[i].equals( "+")) B[i]=B[i-1]+1;
        if (A[i].equals( "-")) B[i]=B[i-1]-1;
    }
    for(i=0;i<B.length;i++)
        System.out.print(B[i]+" ");
    System.out.println();
    return B;
}

public static boolean MutalWord(String s1,String s2,String delim){

```

בודק אם יש מילה משותפת

```
String[] A1=s1.split(delim);
String[] A2=s2.split(delim);
int i,j;
for(i=0;i<A1.length;i++)
    for(j=0;j<A2.length;j++)
        if(A1[i].equals(A2[j]))
            return true;
return false;
}
public static String BiggestString(String[][] P){
    מחזירה את המילה הארוכה וגדולה לקסגורפית במערך 2D ממדי
    int max=0;
    String Word="";
    for(int i=0;i<P.length;i++){
        for(int j=0;j<P[i].length;j++){
            if(max==P[i][j].length() && Word.compareTo(P[i][j])>0){
                Word=P[i][j];
            }
            if(P[i][j]!=null && P[i][j].length()>max){
                max=P[i][j].length();
                Word=P[i][j];
            }
        }
    }
    return Word;
}
public static int CountStr(String s1,String s2){
    מחזיר את מספר ההופעות של המחרוזת המחרוזת השניה בראשונה//
    int cnt=0,temp=0;
    if(s1.indexOf(s2)>0){
        temp=s1.indexOf(s2);
        cnt++;
    }
    while(s1.indexOf(s2,temp+1)>0){
        cnt++;
        temp=s1.indexOf(s2,temp+1);
    }
    return cnt;
}
public static String MoveXtoEnd(String str){
    מזיז את כל ההופעות של איקס קטן לקצה המערך
    int cnt=0;
    for(int i=0;i<str.length();i++){
        if(str.charAt(i)=='x')
            cnt++;
    }
    str=str.replaceAll("x", "");
    for(int i=0;i<cnt;i++){
        str=str+"x";
    }
    return str;
}
public static String mostRepeatngWord(String s){
```

מוצא את המילה שחוזרת הכי הרבה פעמים במחרוזת

```
String[] A=s.split(" ");
int Max=0;
String Word=A[0];
int i,j;
for(i=1;i<A.length;i++)
    if(A[0].equals(A[i]))
        Max++;
int ct=0;
for(i=1;i<A.length;i++){
    ct=0;
    for(j=0;j<A.length;j++)
        if(A[i].equals(A[j]))
            ct++;
    if(ct>Max){
        Max=ct;
        Word=A[i];
    }
}
return Word;
}
public static String NoDuplicateChar(String str){
    מחזיר את המחרוזת בלי תו שחוזר על עצמי
    StringBuffer n=new StringBuffer();
    if(str.charAt(0)!=str.charAt(1))
        n.append(str.charAt(0));
    for(int i=0;i<str.length()-1;i++)
        if(str.charAt(i+1)!=str.charAt(i))
            n.append(str.charAt(i+1));
    return n.toString();
}
public static String DuplicateCharsNoKohavit(String str){
    //עצמם על חזורים התווים וכל כוכביות בלי המחרוזת את מחזיר
    StringBuffer n=new StringBuffer();
    for(int i=0;i<str.length();i++)
        if(str.charAt(i)!='*')
        {
            n.append(str.charAt(i));
            n.append(str.charAt(i));
        }
    return n.toString();
}
public static int longestString(String str){
    מחזיר את המילה הארוכה ביותר מבין המילים במחרוזת כאשר הם מופרדים ע"י שלוש כוכביות
    String[] A=str.split("\\*\\*\\*");
    int max=0;
    for(int i=0;i<A.length;i++)
        max=Math.max(max, A[i].length());
    return max;
}
public static int SumNums(String Str){
    סכום המספרים שמופיעים במחרוזת
    int i,sum,start=0;
```

```

int cnt=0;
sum=0;
i=0;
while(i<Str.length())
{
    cnt=0;
    if(Character.isDigit(Str.charAt(i)) && i<Str.length()){
        start=i;
        cnt++;
        i++;
        while(i<Str.length() && Character.isDigit(Str.charAt(i))){
            i++;
            cnt++;
        }
    }
    i++;
    if(cnt!=0)
        sum+=Integer.parseInt(Str.substring(start,cnt));
}
return sum;
}
public static String Scrambel_ezer(String S){
    char[] e=S.toCharArray();
    int n=S.length();
    int temp,temp2;
    char tmp;
    boolean t=false;
    do{
        temp=1+(int)(Math.random()*n-2);
        temp2=1+(int)(Math.random()*n-2);
        if(temp!=temp2){
            t=true;
            tmp=e[temp];
            e[temp]=e[temp2];
            e[temp2]=tmp;
        }
    }
    while(!t);
    return new String(e);
}
public static String scrambel(String s){
    String[] s1=s.split(" ");
    StringBuffer r=new StringBuffer();
    for(int i=0;i<s1.length;i++)
    {
        r.append(Scrambel_ezer(s1[i])+" ");
    }
    return r.toString();
}
public static String withoutString(String base, String remove){
    return base.replaceAll(remove, "");
}
public static int sumCharDig(String s){

```

מערבל את המילים

מחזיר את סכום הספרות במחרוזת

```
String tmp=s.replaceAll("[a-z]|[A-Z]", "0");
char[] t=tmp.toCharArray();
int sum=0;
for(int i=0;i<t.length;i++){
    sum+=Character.getNumericValue(t[i]);
}
return sum;
}
public static int countTriple(String str) {
    סופר את כמות ההופעות של שלוש תווים רצופים
    int ct=0;
    for(int i=1;i<str.length()-1;i++){
        if(str.charAt(i-1)==str.charAt(i) &&
str.charAt(i)==str.charAt(i+1))
            ct++;
    }
    return ct;
}
public static String mirrorEnds(String st) {
    //מחרוזת המשותף הקצה שהיא מחרוזת מחזורי
    StringBuffer l=new StringBuffer();
    for(int i=0;i<st.length();i++)
        if(st.charAt(i)==st.charAt(st.length()-i-1))
            l.append(st.charAt(i));
        else
            break;
    return l.toString();
}
}
```

Recursion:

```
public static int DifMaxMin(int[] A,int n){
    מחזיר את ההפרש בין המספר הגדול לקטן במערך
    if(n<0) return 0;
    if(n==0) return A[A.length-1]-A[0];
    else
    {
        DifMaxMin(A, n-1);
        if(A[n-1]>A[n]){
            int temp=A[n];
            A[n]=A[n-1];
            A[n-1]=temp;
        }
        DifMaxMin(A, n-1);
    }
    return DifMaxMin(A, n-1);
}
}
public static int ZigZag(int[] A,int l,int r,int k){
    לשים לב לא עובד
    if(k<l || k>r) return 0;
```

```

    if(r==1) return 1;
    if (r==l+1){
        if(A[k]!=A[k+1])
            return 1;
    }
    if(k>l && k<r){
        if((A[k]>A[k-1] && A[k]>A[k+1]) || (A[k]<A[k-1] && A[k]<A[k+1]))
            return 1+ZigZag(A,l,k-1,k-1)+ZigZag(A, k+1, r, k+1);
    }
    else{
        if(k==l)
            return ZigZag(A, k+1, r, k+1);
        if(k==r)
            return ZigZag(A, l, k-1, k-1);
    }
    return 0;
}
public static boolean maze(int[][] A,int i,int j,int n,int m){
    //מבוקש
    if(i==n && j==m) return true;
    if(i<0 || j<0 || A.length<i ||j>A[0].length )
        return false;
    if(A[i][j]==1)
    {
        return maze(A,i+1,j,n,m) || maze(A,i+1,j+1,n,m) ||
maze(A,i,j+1,n,m);
    }
    else
    {
        return false;
    }
}
public static int SumNumDivdedBy3(int[] A,int n){
    if(n<0)
        return 0;
    if(n==0){
        if(A[n]%3==0) return A[n];
    }
    else{
        if(A[n]%3==0)
            return SumNumDivdedBy3(A, n-1)+A[n];
        else
            return SumNumDivdedBy3(A, n-1);
    }
    return 0;
}
public static int floodOnes(int[] A,int l,int r,int k){
    int value=0;
    if(k<l || k> r) return 0;
    if (r==l) return A[k];
    if(A[k]==0)return 0;
    else

```

סכום המפרים המתחלקים בשלוש

floodones גרסה שלי

```

{
    if(k<r)
        value+=floodOnes(A, k+1, r, k+1);
    if(k>l)
        value+=floodOnes(A, l, k-1, k-1);
    return value+1;
}
}
public static int RezfdevidedBy3(int [] A,int l, int r, int k)
    רצף הארוך ביותר של מספרים המתחלקים בשלוש שכולל מיקום מסויים במערך
{
    int sum=0;
    if(k<l || k>r) return 0;
    if(r==l){
        if( A[l]%3==0)
            return 1;
        else return 0;
    }
    if(A[k]%3==0){
        if(k>l)
            sum+=RezfdevidedBy3(A, l, k-1, k-1);
        if(k<r)
            sum+=RezfdevidedBy3(A, k+1, r, k+1);
        return sum+1;
    }
    else
        return 0;
}
public static int howManyApearance(int n,int k){
    מחזיר את מספר ההופעות של ספרה במספר
    if(n>0 && n<10){
        if(n==k)
            return 1;
        else return 0;
    }
    else
    {
        if (n%10==k)
            return howManyApearance(n/10, k)+1;
        else
            return howManyApearance(n/10, k);
    }
}
public static String reverseAndRemoveDot(String s){
    מחזיר את המחרוזת בסדר הפוך בלי נקודה
    if(s.length()==1)
    {
        if(s.charAt(0)!='.')
            return s;
        else
            return "";
    }
    else{

```



```

        if(s.charAt(0)!='.')
            return reverseAndRemoveDot(s.substring(1))+s.charAt(0);
        else
            return reverseAndRemoveDot(s.substring(1));
    }
}
public static boolean check(int A[],int n,int k){
    if (k<0)return false;
    if(k==0) return A[0]==n*-1;
    if(A[k]==-1*n) return true;
    else
        return check(A,n,k-1);
}
public static boolean negative(int[] A){
    for(int i=0;i<A.length;i++){
        if (!check(A,A[i],A.length-1))
            return false;
    }
    return true;
}
public static void captilizer(String s,int n){
    כל התמורות של אותיות גדולות וקטנות כך שסדר המילה נשמר
    if(n<0)
        System.out.println(s);
    else{
        char[] a=s.toCharArray();
        a[n]-= 32;
        captilizer(s, n-1);
        captilizer(new String(a), n-1);
    }
}
public static void PrintTriange(int n){
    הדפסת שתי משלושים בעלי קודקוד משותף בסיס בגודל n
    if(n!=0)
    {
        for(int i=0;i<n;i++)
            System.out.print('*');
        System.out.println();
        PrintTriange(n-1);
        for(int i=0;i<n;i++)
            System.out.print('*');
        System.out.println();
    }
    if(n==0)
        System.out.println("-----zero-----");
}
public static void PrintNumber(int n){
    מדפיס את המספר כך שהוא מתחיל מהמספר וכל פעם מוריד את ספרת האחדות
    if(n>=0 && n<10)
        System.out.println(n);
    else{
        System.out.println(n);
    }
}

```

```

        PrintNumber(n/10);
    }
}
public static int matchingDig(int n, int m){

```

מחזיר את מספר הספרות התואמות

```

    if(n>=0 && n<10 && m>=0 && m<10)
        if(n==m)
            return 1;
        else
            return 0;
    else
        if (n%10 == m%10)
            return matchingDig(n/10, m/10)+1;
        else return matchingDig(n/10, m/10);
}

```

```

public static void print(char[] A){
    for(int i=0;i<A.length;i++)
        System.out.print(A[i]);
    System.out.println();
}

```

```

public static void anagram(char[] A,int n){

```

מדפיס מחרוזת כך שיופיעו כל התמורות האפשרויות מהמחרוזת יש לשלוח את המחרוזת כמערך של תוים

```

    char temp,temp2;
    for(int i=n;i<A.length;i++){
        temp=A[i];
        A[i]=A[n];
        A[n]=temp;
        anagram(A, n+1);
        temp2=A[i];
        A[i]=A[n];
        A[n]=temp2;
    }
    if(n==A.length-1) print (A);
}

```

```

public static int recrPascal(int row, int col)
{

```

```

    int val1, val2, result = 0;
    if (row == 0 || col == 0 || row == col + 1)
    {
        return 1;
    }

```

```

    val1 = recrPascal(row - 1, col - 1);
    val2 = recrPascal(row - 1, col);

```

```

    return val1 + val2;
}

```

```

public static void printPascalRecursion(int maxRows)

```

משולש פסקל

```

{
    for (int i = 0; i <= maxRows; i++)
    {
        for (int j = 0; j < i; j++)

```

```

    {
        System.out.print(recrPascal(i, j) + " ");
    }
    System.out.println();
}
}

```

```

public static boolean groupSum(int start, int[] nums, int target) {
    מחזיר אם סכום חלק מהמערך שווה למטרה
    // Base case: if there are no numbers left, then there is a
    // solution only if target is 0.
    if (start >= nums.length) return (target == 0);

    // Key idea: nums[start] is chosen or it is not.
    // Deal with nums[start], letting recursion
    // deal with all the rest of the array.

    // Recursive call trying the case that nums[start] is chosen --
    // subtract it from target in the call.
    if (groupSum(start + 1, nums, target - nums[start])) return true;

    // Recursive call trying the case that nums[start] is not chosen.
    if (groupSum(start + 1, nums, target)) return true;

    // If neither of the above worked, it's not possible.
    return false;
}

```

```

public static boolean groupSumNoAdj(int start, int[] nums, int target) {
    מחזיר אם סכום חלק מהמערך שווה למטרה מספק עוקב למספר שנבחר לא יכול להבחר

    if (start+1 >= nums.length) return (target == 0);

    if (groupSumNoAdj(start + 1, nums, target - nums[start]-
nums[start+1])) return true;

    if (groupSumNoAdj(start + 1, nums, target-nums[start+1])) return true;

    return false;
}

```

```

public static int cTriples (int[] A, int n) {
    כמות השלושות העוקבות במספר

    int tmp=0;
    if(n<2)
        return 0;
    else{
        if(A[n-2]<A[n-1] && A[n-1]<A[n]) tmp++;
        return cTriples(A,n-1)+tmp;
    }
}

```