עבודה שלישית – מסדי נתונים

שאלה 1:

סעיף 1:

```
db.createCollection("books",{ capped : true, size : 3})
```

אופציה ראשונה להכנסה:

```
db.books.insert(
{
  name: "Book1",
  author: "a1",
  Publisher: "p1",
  YearOfPublishing:"2018",
  Content: "Once upon a time"
})

db.books.insert(
{
  name: "Book2",
  author: "a2",
  Publisher: "p2",
  YearOfPublishing:"2019",
  Content: "Good luck in the chore"
})


db.books.insert(
{
  name: "Book3",
  author: "a3",
  Publisher: "p3",
  YearOfPublishing:"2020",
  Content: "the third book"
})
```

אופציה שניה להכנסה:

```
db.books.insertMany(
[
{ name: "Book1",
  author: "a1",
  Publisher: "p1",
  YearOfPublishing:"2018",
  Content: "Once upon a time"
},
{
  name: "Book2",
  author: "a2",
  Publisher: "p2",
  YearOfPublishing:"2019",
  Content: "Good luck in the chore"
},
{
  name: "Book3",
  author: "a3",
  Publisher: "p3",
  YearOfPublishing:"2020",
```

```
  Content: "the third book"
}
]);
```

סעיף שני:

```
var mapper=function(){
var arr = this.Content.split(' ');
for(i in arr){
        emit(arr[i].length,1);
    }
};

var reducer=function(keyLen,value){
var counter=0;
for(var i=0;i<value.length;i++){
        counter++;
}
return counter;
};

db.books.mapReduce(
mapper,
reducer,
{
        out:"ans1"
}
);

db.ans1.find()
```

שאלה 2:

```
MATCH(g:Person),(f:Person) WHERE g.name="GAL" AND f.name<>"GAL" AND
(f)-[:FriendOf*..3]->(g) with(f)
MATCH(g:Person),(m:Movie) WHERE g.name="GAL" AND (g)-[:Like]->(m) AND
(g)-[:Watch]->(m)  AND (f)-[:Like|:Watch]->(m)
with(m),COUNT(DISTINCT f) AS cf
where cf>=2
return DISTINCT m
```

שאלה 3:

```
let $listOfCalories := doc("breakfast_menu.xml")//calories
let $avgCalories := avg($listOfCalories)
for $x in  doc("breakfast_menu.xml")//food where $x/calories >
$avgCalories return $x/name
```

שאלה 4:

באופן פשוט ניתן לעשות:

```
Stream.of(0,95,107,-8,83,74,100,67,54,98)
.filter(num->num>=60&&num<=100)
.map(grade->100-grade).forEach(System.out::println);
```

ניתן לעשות גם בתור פונקציה:

```
public static void q4(int collection[]) {
Arrays.stream(collection).filter(num->num>=60&&num<=100)
.map(grade->100-grade).forEach(System.out::println);
}
```

שאלה 5:

Subject	Predicate	Object
111	name	ישראל ישראל
111	age	15
111	Father_ID	444
222	name	פלוני אלמוני
222	age	2
222	Father_ID	333
333	name	ג'ון סמית
333	age	30
333	Father_ID	444
444	name	ראובן אריאל
444	age	81
444	Father_ID	555

להלן השאילתא שתחזיר את על הנכדים של ראובן אריאל:

```
SELECT ?nameToReturn where{
?idReuben name "ראובן אריאל".
?idChild Father_ID ?idReuben.
?idGrandSon Father_ID ?idChild.
?idGrandSon  name ?nameToReturn.
}
```

שאלה 6:

סעיף 1:

שלב ראשון – נדרג את המילים על מנת לראות מי מופיעה הכי פחות ולכן היא הכי "שווה".

איזה =0 (בגלל שהיא מופיעה 0 פעמים היא לא רלוונטית), הבוקר=2 (ולכן הכי "שווה"),צבע = 3,השמיים=4 (מופיעה הכי הרבה פעמים ולכן הכי פחות "שווה").

יש לנו שלושה משפטים באורך 5,משפטים 2,3,4 במשפט 2 מופיעות שלושת המילים השוות והוא גם באורך הכי קצר ולכן מפשט 2 הכי "שווה".

נשארנו עם משפט 3,4 בשני המשפטים מופיעות 2 מילים שוות – בשתיהן יש את המילה "השמיים" ואילו המילה השנייה במשפט 3 יותר "שווה" מהמילה השנייה במשפט 4. ולכן משפט 4 "שווה" יותר ממשפט 3.

לכן נשארנו עם משפט 4 ו1 שבשניהם יש את אותם המילים אבל משפט 1 יותר ארוך ולכן "שווה" פחות.

ולכן הדירוג הוא:

1.משפט 2

2.משפט 3

3.משפט 4

4.משפט 1

סעיף 2:

	איזה	צבע	השמיים	הבוקר	מס' המילים במשפט
משפט 1	0	1	1	0	9
משפט 2	0	1	1	1	5
משפט 3	0	0	1	1	5
משפט 4	0	1	1	0	5
	0	3	4	2	

TF-IDF (משפט 1): 1/9*log(4/3)+1/9*log(4/4) =0.013882

TF-IDF (משפט 2): 1/5*log(4/3)+1/5*log(4/4)+1/5*log(4/2)= 0.085193

TF-IDF (משפט 3): 1/5*log(4/4)+1/5*log(4/2)= 0.060205

TF-IDF (משפט 4): 1/5*log(4/3)+1/5*log(4/4)= 0.0249877

ולכן הדירוג הוא:

1.משפט 2

2.משפט 3

3.משפט 4

4.משפט 1