

[דף סיכום בחינה](#)

מזהה בחינה: 000000090648 מזהה סטודנט: 0212866586

מזהה קורס: 7217-180 שם קורס: DB Mongo

מספר שאלה	הערה	ניקוד מירבי	ציון
1.1		3.00	2.00
1.2		3.00	3.00
1.3		5.00	2.00
1.4		5.00	3.00
1.5		5.00	5.00
1.6		5.00	5.00
1.7		7.00	0.00
1.8		7.00	0.00
1.9		9.00	5.00
1.10		11.00	0.00
2.1		4.00	0.00
2.2		4.00	0.00
2.3		5.00	0.00
2.4		7.00	0.00
3	הניקוד מאוד מתחשב..	20.00	15.00

ציון בחינה סופי : 40.00

הבחינה הבדוקה בעמודים הבאים

בית הספר הארצי להנדסאים (ע"ר)

מחברת בחינה

הוראות לנבחן

(7)

ID 0212866586

Exam 000000090648

1. לפני תחילת הבחינה אנא קרא בעיון את ההוראות ומלא את הפרטים בכתב יד ברור.
שים לב: מחברות הבחינה נסרקות למאגר נתונים.
יש להקפיד: לא לקפל, לא לתלוש ולא לכתוב בצבעים או בעפרון.
2. כתוב את פרטייך האישיים על גבי מחברת הבחינה מיד עם תחילת הבחינה.
3. כתוב את תשובותיך בכתב יד ברור **משני צידי הדף.**
4. ציין בראש כל דף את מספר השאלה עליה אתה עונה.
5. כתוב את כל עבודתך (כולל הטייטה) במחברת זו. אל תכתוב דבר על דף אחר.
6. מתח קו אלכסוני ורשום "מבוטל" על אותם חלקים שאינך רוצה שייבדקו.
7. אם מחברת זו לא תספיק לך, בקש מהמשיגי מחברת נוספת. כתוב על כל מחברת את פרטייך האישיים.
8. הוצאת מחברת בחינה (או חלק ממנה) מחוץ לחדר הבחינה הינה עבירה משמעתית חמורה.
9. על הנבחן לציית להוראות המשיגים. כל פנייה למשיגי תיעשה על ידי הרמת יד בלבד.
10. נבחן שיפריע למהלך הבחינה, יועמד לדין משמעתי של ביה"ס. ניתן להפסיק את בחינתו ולתבעו בוועדת משמעת של ביה"ס.
11. נבחן לא יהיה רשאי לעזוב את חדר הבחינה, אלא על פי נוהלי בית הספר ובאישור המשיגי.
12. מותר להשתמש בחומר עזר רק אם הדבר הותר במפורש בכתב בגוף שאלון הבחינה. הימצאות חומר אסור בידי הנבחן בשעת הבחינה על שולחנו, או בסמוך אליו, מהווה עבירת משמעת חמורה.

הערה: ההוראות מתייחסות לזכר ולנקבה כאחד.

תאריך הבדיקה: _____

אני מאשר כי קראתי את ההוראות האמורות לעיל: _____
חתימת המרצה: _____
חתימת הסטודנט

ציון:

בהצלחה!

לתשומת לבך:

יש לכתוב את תשובותיך משני צידי הדף
אין לכתוב מעבר לקו האדום
יש לכתוב את הבחינה בעט (כחול/שחור) בלבד.

שם משפחה: קיאלי שם פרטי: מכאל

מספר תעודת זהות: 8 1 8 8 6 6 5 8 6

תאריך הבחינה: 16/8/24 מגמה: מכא מחזור: 2

שם הקורס: מסמט שם המרצה: משה

סמסטר: חורף / אביב / קיץ מועד: אמצע / א / ב / מיוחד

מחברת מספר מתוך סה"כ מחברות

קורס MongoDB

ההנחיות בשאלון זה מנוסחות בלשון זכר ומכוונות לנבחנות ולנבחנים כאחד.

בסיום המבחן חובה למסור למשגיחים את:

- א. מחברת הבחינה.
- ב. טופס הבחינה, עליו נכתב בבירור שם ות.ז. של הסטודנט.
- ג. שני דפי העזר שהביא הסטודנט למבחן (סעיף ג' בהמשך).

א. משך המבחן: שלוש שעות.

ב. מבנה השאלון ומפתח הערכה: יש לענות על כל השאלות. הניקוד כתוב בגוף המבחן.

ג. חומר עזר מותר בשימוש:

שני דפי עזר, בגודל A4, שהינם דפי עזר אישיים.
חובה לרשום את השם ות.ז. על הדפים האלו, ובסיום המבחן למסור אותם למשגיחים.

ד. הוראות מיוחדות:

- 1. כתב יד שאיננו ברור - לא ייבדק, או שיירד ניקוד משמעותי בגין כך!!
- 2. חל איסור מוחלט להוציא שאלון או מחברת בחינה מחדר הבחינה במהלך המבחן.
- 3. במידה ועל דפי הנוסחאות האישיים לא יופיעו פרטי הסטודנט - הדפים יוחרמו ולא יוכלו לשמש את הסטודנט במהלך המבחן.
- 4. יש לכתוב בעט בלבד.
- 5. אם נכתבה טיוטה במחברת הבחינה - יש למחוק אותה ע"י העברת קו כדי שלא תיבדק.
- 6. בשאלות בהן נדרש להסביר - חובה לנמק.
- 7. בשאלות שכאלו, תשובה לא מנומקת - לא תקבל ניקוד.
- 7. התחל לפתור כל שאלה בעמוד חדש במחברת הבחינה. כתוב בבירור את מספר השאלה.

בהצלחה!

ברוך
כיון

1. השאלות בחלק זה יבוססו על מסד נתונים במונגו, המכיל מידע לגבי ספרים בספרייה. להלן דוגמא לאחד המסמכים שנמצא בטבלה שבמסד נתונים זה:

```
{
  "_id": "978-3-16-148410-0",
  "title": "The Great Story of all 48 Classes",
  "description": "The auther presents readers with marvelous characters",
  "cover_type": "Hardcover",
  "author_name": "Lior B",
  "book_dimensions": {
    "width": 15,
    "length": 10,
    "height": 2.5
  },
  "best_seller": true,
  "publication_year": 2024,
  "publisher": "Ben Bassat Publishing Ltd",
  "num_of_pages": 1759,
  "language": "Hebrew",
  "book_type": "Extreme Adventure",
  "cost": 998.87,
  "rating": 10
}
```

כמובן, שלא כל המסמכים חייבים להכיל בתוכם את כל התכונות שלעיל.

יש לקרוא היטב את כל הנדרש בכל שאילתה. כפי שהוסבר בכיתה, לאופן כתיבת

השאילתות יהיה חלק בניקוד שלהן, למשל:

- תחביר נכון. הקפדה על מספר סוגריים מדויק, כתיב מדויק בשמות פקודות, וכו'.
- כל "internal json" חייב להופיע בשורה נפרדת. למשל:

```
db.myCollection.aggregate(
```

```
[
  { $group: ... },
  { $project: ... },
  { $sort: ... },
  { $match: ... }
]
```

```
)
```

- יש לדייק בשמות העמודות, לפי המוצג בדוגמת המסמך שבראש עמוד זה.
- כתב ברור מאוד כפי שמצוין בפתיח המבחן.

יש לכתוב את השאילתות הנדרשות על טבלה בשמכם הפרטי המכילה מסמכים המתארים ספרים כמוראה בעמוד הקודם. להלן השאילתות שעליכם לכתוב:

א. (3 נקודות) כתוב שאילתה המציגה מידע על הספרים המוגדרים כ- best_seller IX כאלו שסוג הספר (book_type) הינו "History". המידע שנדרש להציג לכל אחד מהספרים האלו כולל רק את התכונות הבאות: שם הספר, שם המחבר, תיאור הספר.

ב. (3 נקודות) כתוב שאילתה המחזירה את כמות הספרים שפורסמו בשנת 2023 ואילך (publication_year), וגם שסוג הכריכה שלהם ("cover_type") הינה "Paperback". במילים אחרות, השאילתה עונה על השאלה "כמה ספרים כאלו ישנם". בסעיף זה עליך להשתמש בפקודת find.

ג. (5 נקודות) פתור את השאילתה מהסעיף הקודם באמצעות שימוש בפקודה aggregate תוצאת השאילתה הרצויה בסעיף זה אמורה להציג את התשובה באופן הבא:
[{ 'Total number of books': 3 }]
כלומר יש להציג את התוצאה בשדה ששמו "Total number of books", ורק השדה הזה אמור להיות בפלט המתקבל מהשאילתה, בדיוק כמו שמוצג לעיל.

ד. (5 נקודות) לאחר מתן ציונים לדירוגי הספרים (התכונה rating), הוחלט לעדכן את מחירם של כל הספרים שערך הדירוג שלהם הינו הציון המקסימלי 10. לפי ההחלטה הזו: מחיר הספר יעודכן להיות המקסימום שבין מחיר הספר הנוכחי לבין מחיר 50 ₪.
לדוגמה, אם מחירו של הספר לפני ניסיון העידכון הינו 49 ₪ אז המחיר יעודכן ל 50 ₪, ואילו אם מחירו היה 51 ₪ אז המחיר לא יעודכן וישאר 51 ₪.

המשך בעמוד הבא

ה. (5 נקודות) מבין הספרים שפורסמו עד שנת 2021 (כולל), עבור הספר השלישי בעלותו (לא זה הכי יקר, אלא זה בעל המחיר השלישי בגודלו) - הצג רק את שמו של הספר והסוג שלו (book_type).

ניתן להניח שאין שני ספרים בעלי עלות זהה, וכי ישנם לפחות שלושה ספרים בספרייה. יש לכתוב את השאילתה הנדרשת בסעיף זה באמצעות שימוש בפקודה find .
לדוגמה, נניח שיש 4 ספרים שפורסמו עד שנת 2021:

ספר א' עולה 123 ₪,

ספר ב' עולה 98 ₪,

ספר ג' עולה 78 ₪,

ספר ד' עולה 101 ₪.

במצב כזה הספר השני בגודלו מבחינת מספר העמודים הינו ספר ד' כי לאחר מיון לפי מספר העמודים מקבלים: 123, 101, 98, 78. יש להציג רק את שמו של הספר ואת הסוג שלו.

ו. (5 נקודות) פתור את השאילתה מהסעיף הקודם באמצעות שימוש בפקודה aggregate

ז. (7 נקודות) עבור כל הספרים שאינם כתובים בשפה האנגלית (כלומר לא כתובים באנגלית, אלא בשפה אחרת) - יש להוזיל את מחירם ב 5 ₪ (המחיר הינו הערך שבשדה cost).

לדוגמא: נניח שקיים ספר בשפה האיטלקית שעלותו הינה 100.12 ₪, הפקודה הנדרשת בסעיף זה תתבקש לעדכן את עלותו ל- 95.12 ₪ (כי: $100.12 - 5 = 95.12$). כך צריך לעדכן לכל אותם הספרים שלא כתובים בשפה האנגלית.

ח. (7 נקודות) בספרייה שלנו נכתבו ספרים בכל מיני שפות, ולכן נוכל לחשב את העלות הממוצעת של הספרים שנכתבו בכל שפה.
אם למשל נכתבו שני ספרים בשפה הולנדית, הראשון בעלות 100 ₪ והשני בעלות 200 ₪, הרי שהעלות הממוצעת של ספרים שנכתבו בשפה ההולנדית הינה 150 ₪.
כתוב שאילתה שמטרתה להציג את העלות הממוצעת של הספרים בכל שפה שבה יש ספרים בטבלה שלנו.
הפלט של השאילתה צריך להציג את המידע באופן הבא (הערכים הם להמחשה):

```
[  
  { 'Average cost': 534.43, 'Language Name': 'Hebrew' },  
  { 'Average cost': 36.35727272727273, 'Language Name': 'English' }  
]
```

מהדוגמה שלעיל ניתן להבין למשל כי העלות הממוצעת של ספרים בעברית- הינה 534.43 ₪.
כאמור, השאילתה המבוקשת צריכה להציג את עלות הספרים הממוצעת בכל אחת מהשפות.

ט. (9 נקודות) יש לכתוב שאילתה להצגת השפה שבה נכתבו הכי הרבה ספרים.
תוצאות השאילתה צריכות להציג את שם השפה הזו ואת מספר הספרים שנכתבו בה.

י. (11 נקודות)

נפח ספר מתקבל מתוצאת החישוב של הכפלת ממדיו: $width * length * height$.
כמוראה בהגדרת המסמך, ממדי הספר מופיעים בתכונה: `book_dimensions`
כתוב שאילתה שמטרתה להציג את הנפח של הספר היקר ביותר שנכתב בכל אחת מהשפות.
הפלט הנדרש מהשאילתה צריך להיות ממוין בסדר יורד לפי אותו חישוב נפח, ולהיות מוצג במבנה
שכזה:

```
[  
  { _id: 'Hebrew', 'Volume of the most expensive book': 375 },  
  { _id: 'English', 'Volume of the most expensive book': 228.8 }  
]
```

מהפלט של הדוגמה הנ"ל ניתן להבין למשל שהנפח של הספר היקר ביותר שנכתב בעברית הוא
375.

2. להלן חלק מסקריפט המגדיר מקומות ויחסי טיסות ביניהם:

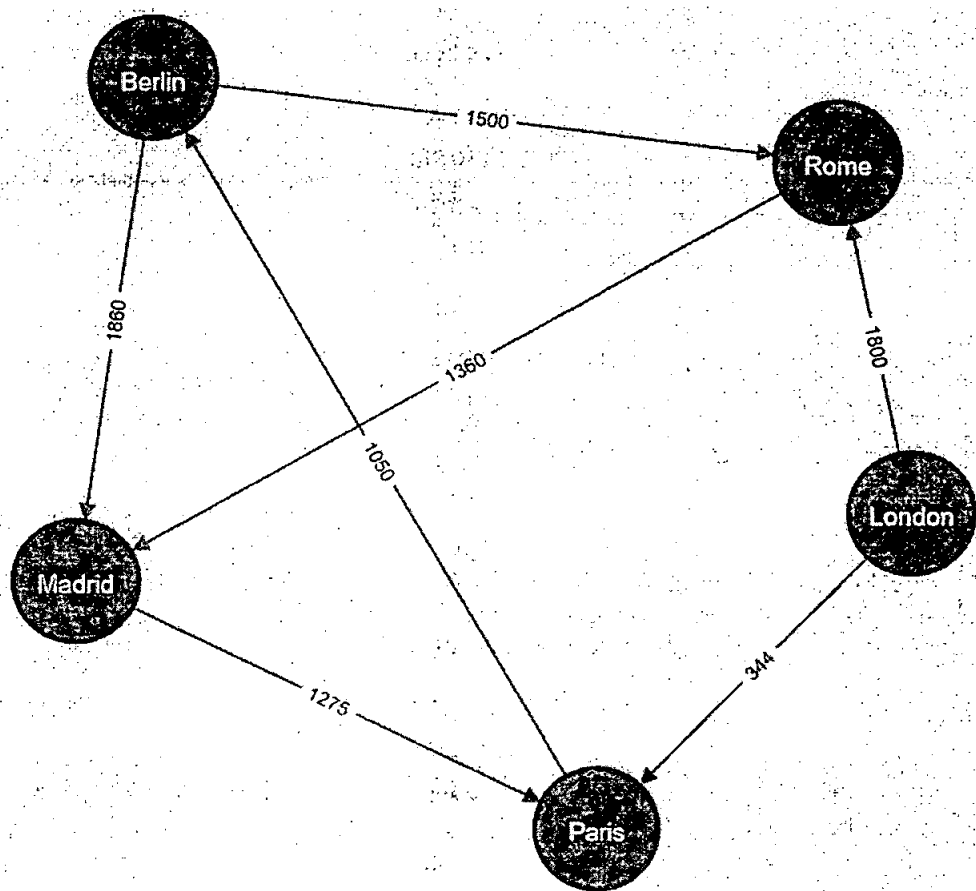
```
// Create city nodes
CREATE (london:City {name: 'London'})
CREATE (paris:City {name: 'Paris'})
CREATE (berlin:City {name: 'Berlin'})
CREATE (rome:City {name: 'Rome'})
CREATE (madrid:City {name: 'Madrid'})

// Create road relationships with distances and additional properties
CREATE (london)-[:Connected {distance: 344, cost:200, speedLimit: 65}]->(paris)
CREATE (paris)-[:Connected {distance: 1050, cost:210, speedLimit: 70}]->(berlin)
CREATE (berlin)-[:Connected {distance: 1500, cost:220, speedLimit: 130}]->(rome)
CREATE (rome)-[:Connected {distance: 1360, cost:230, speedLimit: 130}]->(madrid)
CREATE (madrid)-[:Connected {distance: 1275, cost:240, speedLimit: 60}]->(paris)
CREATE (london)-[:Connected {distance: 1800, cost:250, speedLimit: 130}]->(rome)
CREATE (berlin)-[:Connected {distance: 1860, cost:250, speedLimit: 130}]->(madrid)
```

הרצת הסקריפט מייצרת גרף המציג מידע על דרכים בין הערים הנתונות (מידע שאיננו אמיתי).

יתכן בהחלט שיש יותר ממסלול אחד בין שתי ערים.

- א. (4 נקודות) כתבו סקריפט להצגת מספר הערים המופיעות במסלול הקצר ביותר (מבחינת מספר הקשתות) שבין הערים London - Madrid, ובהנחה שהמסלול הזה מונה לכל היותר 10 קשתות. אדגיש: לא מדובר במסלול הכי קצר מבחינת ה- distance שמוגדר על רכיביו, אלא במסלול הכי קצר מבחינת מספר הקשתות שמרכיבות אותו, והשאלית נדרשת להציג את מספר הערים שבמסלול.
- ב. (4 נקודות) כתבו סקריפט להצגת סך העלות הכוללת של המסלול שהתקבל בסעיף ב' (כלומר סך כל העלויות של כל קטעי המסלול הזה).
- ג. (5 נקודות) כתבו סקריפט להצגת שמות הערים שמן ניתן להגיע בנסיעה של שני שלבים אל העיר מדריד (העיר מדריד היא היעד).
- למשל, עבור המפה המוגדרת בעמוד הבא, ניתן לראות שהערים מהן ניתן להגיע למדריד בנסיעה בת שני שלבים הן: פריז, לונדון וברלין.
- לא ניתן למשל להגיע למדריד בנסיעה בת שני שלבים היוצאת מרומא.
- ד. (7 נקודות) הצג את שמות אותן הערים המקיימות את התנאים הבאים:
 - a. יש ביניהם קשת ישירה לעיר מדריד.
 - b. המרחק על הקשת הזו קטן מ 2000
 - c. שמות הערים האלו מתחיל באות B



המשך בעמוד הבא


```

<!DOCTYPE Company [
  <!ELEMENT Company (companyName,
                      managerName,
                      employees
                      )>

  <!ELEMENT companyName (#PCDATA)>
  <!ELEMENT managerName (#PCDATA)>
  <!ELEMENT employees (employee+)>

  <!ELEMENT employee (empName,salary, empAddress+)>
  <!ELEMENT empName (last, middle?, first)>
  <!ELEMENT first (#PCDATA)>
  <!ELEMENT middle (#PCDATA)>
  <!ELEMENT last (#PCDATA)>
  <!ELEMENT salary (amount, currency)>
  <!ELEMENT amount (#PCDATA)>
  <!ELEMENT currency (#PCDATA)>
  <!ELEMENT empAddress (#PCDATA | city | street | houseNum)*>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT houseNum (#PCDATA)>

  <!ATTLIST Company companyId ID #REQUIRED>
  <!ATTLIST employee empId ID #REQUIRED>
  <!ATTLIST employee managerId IDREF #IMPLIED>
  <!ATTLIST employee type (FullTime | Student ) #REQUIRED>
  <!ATTLIST employee managerOf IDREFS #IMPLIED>
]>

```

הסבר לגבי התכונה managerId שבאלמנט employee: בתכונה זו יש את ה-ID של המנהל של אותו

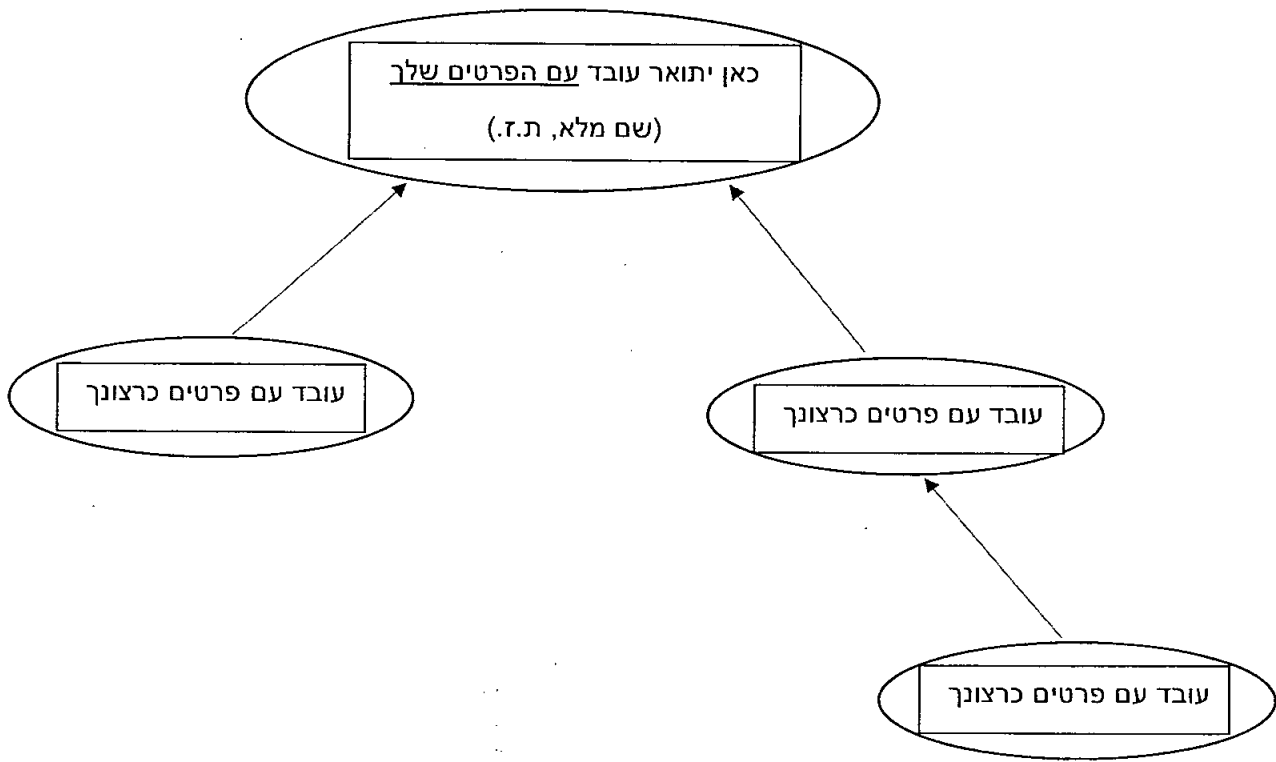
employee.

הסבר לגבי התכונה managerOf שבאלמנט employee: בתכונה זו יש את כל ה-ID של כל עובדים

שכפופים לאותו employee.

המשך בעמוד הבא

20 נקודות) בהתאם ל- DTD שלעיל- כתבו XML המכיל 4 עובדים במבנה הארגוני הבא, כך שהוא יקיים את הדרישות המיוחדות בהמשך עמוד זה:

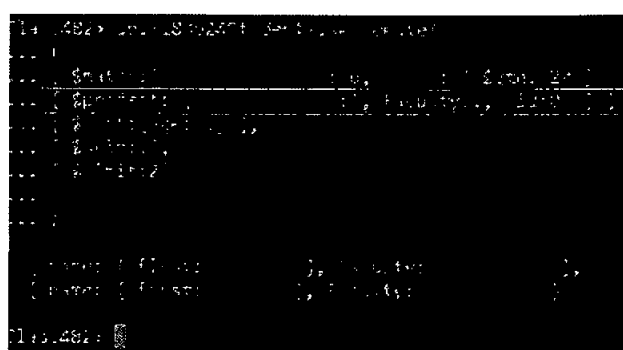


דרישות מיוחדות בכתיבת ה- XML :

- אתה מוגדר כמנהל של הארגון ששורטט לעיל, ולכן עליך להכניס במקומות המתאימים את ת.ז. שלך ואת שמך המלא.
- הקפד ששניים מתוך ארבעת העובדים בשרטוט הנ"ל- יוגדרו במשרת סטודנט, והשניים האחרים יוגדרו כעובדים במשרה מלאה (FullTime)
- הקפד להגדיר כל כתובת (empAddress) במבנה אחר, לפי המותר ב- DTD.

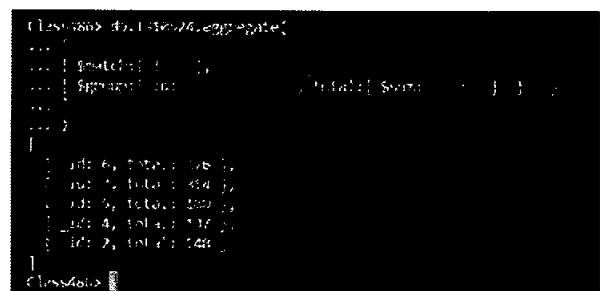
**בהצלחה,
ליאור.**

נמצאים בשימוש נכון, לפי ה-DTD. לדוגמה, האלמנט middleName צריך להיות תקין אם הוא מופיע, גם אם הוא אופציונלי.

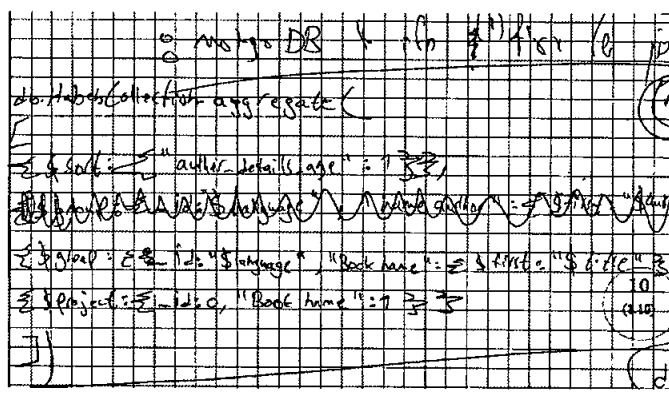


ag- מבצע סדרה של פעולות:

- אשונה אנו מבקשים רק אוכלוסייה מסוימת.
 - ידיה נבקש לקבל רק שדות מסוימים.
 - ילישית נבקש למיין לפי גיל בסדר יורד.
 - ביעית נדגל על המבוגר ביותר (לפי דרישת התרגיל).
 - מישית נגביל את התוצאות לשתיים בלבד (לפי דרישת התרגיל).
- לראות שקיבלנו את אותן התוצאות כמו בהרצה של ה find.



- Note that in the above example:
- (*) The empty match criteria indicates that we are observing all the document.
 - (*) We are grouping according to the value of the semester number. Meaning grouping is based on the SemesterNum column.
 - (*) We calculate the sum of ages of all the students in each group.
 - (*) The calculation result is presented under the column name "total" (in the coming example we will see how to change that name).



```
<day>13</day>
<month>July</month>
<year>2024</year>
</date>
<items>
<item serialNum="5004" discountedItems="5006">
<itemDesc>Laptop</itemDesc>
<itemPrice>1200</itemPrice>
<quantity>1</quantity>
</item>
<item serialNum="5005">
<itemDesc>Mouse</itemDesc>
<itemPrice>20</itemPrice>
<quantity>2</quantity>
</item>
</items>
</order>
</orders>
```

במשימה ב' נתבקשת למצוא 5 שגיאות בקובץ ה-XML שנוצר בעמוד הקודם. הנה 5 השגיאות האפשריות:

- היעדר שדה middleName (שם אמצעי): לפי ה-DTD, צריך להיות אלמנט middleName בתוך custName. בקובץ ה-XML השדה הזה קיים אך ריק, וזה עשוי להיחשב כשגיאה אם שם אמצעי נדרש או שהשדה לא אמור להיות ריק.
- מבנה כתובת לא עקבי: בקובץ ה-XML נעשה שימוש באלמנט houseNum בתוך address, שמופיע גם ב-DTD. יש לוודא שהשימוש באלמנט הזה ובמבנה הכולל של הכתובת הוא עקבי ומתאים למפרט ב-DTD.
- ערך לא תקין עבור אמצעי התשלום: לפי ה-DTD, הערכים החוקיים לאטריביוט paymentMethod הם "Cash", "CreditCard" או "Points". בקובץ ה-XML מופיע הערך "MyPoints", שאינו תואם את הערכים שהוגדרו ב-DTD.
- שימוש באטריביוט discountedItems: לפי ה-DTD, האטריביוט discountedItems באלמנט item הוא לא חובה (IMPLIED#). יש לוודא שהוא נמצא בשימוש נכון ואם לא נדרש, יש להסירו.
- אלמנטים ואטריביוטים אופציונליים לא נכונים/שגויים: יש לוודא שכל האלמנטים והאטריביוטים האופציונליים

CREATE (chicago)-[:FLIGHT {distance: 802, duration: 130, pilot: 'Linda Clark'}]->(dallas)
CREATE (cleveland)-[:FLIGHT {distance: 462, duration: 95, pilot: 'Chris Johnson'}]->(nyc)
CREATE (nyc)-[:FLIGHT {distance: 1389, duration: 210, pilot: 'Chris Johnson'}]->(dallas)
CREATE (la)-[:FLIGHT {distance: 1745, duration: 270, pilot: 'David Kim'}]->(chicago)
Here are the queries to answer each of the tasks from the image:

Task 1: Find the Two Longest Flights by Distance

MATCH (c1:City)-[:FLIGHT]->(c2:City)
RETURN c1.name AS from, c2.name AS to, f.distance
ORDER BY f.distance DESC
LIMIT 2

Task 2: Find the Shortest Flight and Return Its Distance

MATCH (c1:City)-[:FLIGHT]->(c2:City)
RETURN c1.name AS from, c2.name AS to, f.distance
ORDER BY f.distance ASC
LIMIT 1

Task 3: Return the Total Number of Flight Routes

MATCH (c1:City)-[:FLIGHT]->(c2:City)
RETURN COUNT(*) AS total_flight_routes

Task 4: Find the Shortest Flight from Los Angeles to Dallas

MATCH (la:City {name: 'Los Angeles'})-[:FLIGHT]->(dallas:City {name: 'Dallas'})
RETURN f.distance AS distance
ORDER BY f.distance ASC
LIMIT 1

Task 5: Find the Total Number of Flights Piloted by "Chris Johnson"

MATCH (c1:City)-[:FLIGHT {pilot: 'Chris Johnson'}]->(c2:City)
RETURN COUNT(*) AS chris_johnson_flights

Task 6: Find the Total Number of Flight Route Distance Greater Than 100

MATCH (c1:City)-[:FLIGHT]->(c2:City)
WHERE f.distance > 100
RETURN COUNT(*) AS long_distance_routes

Task 7: Return the Names of All Pilots Who Piloted a Flight with a Duration Greater Than 200

WHERE f.duration > 200
RETURN DISTINCT f.pilot

Task A: Create an XML Document Based on the Provided Structure

The XML document should adhere to the DTD specified on the first page. It should contain 5 elements (or "fields") that match the provided structure. Based on the DTD, a valid XML document might look like this:

<order transactionId="12345" paymentMethod="MyCard">
<customer customerId="C1234">
<firstName>Jane</firstName>
<middleName></middleName>
<lastName>Doe</lastName>
</customer>
<address>
<city>Los Angeles</city>
<street>123 Main Street</street>
<houseNum>456</houseNum>
</address>
</order>
</xml>

\$skip •

```
db.orders.aggregate([
  { $sort: { date: -1 } },
  { $skip: 10 }
]);
```

3. Indexing

```
createIndex(keys, options) •
db.users.createIndex({ name: 1 });
dropIndex(indexName) •
db.users.dropIndex("name_1");
listIndexes() •
db.users.listIndexes();
dropIndexes() •
db.users.dropIndexes();
```

6. Data Validation

```
createCollection(name, options) •
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "email"],
      properties: {
        name: {
          bsonType: "string",
          description: "must be a string and is required"
        },
        email: {
          bsonType: "string",
          pattern: "^.+@.+\\.+$",
          description: "must be a string and match the email format"
        }
      }
    }
  }
});
```

7. Utility M

```
renameCollection(newName, options)
db.users.renameCollection("custo
drop()
db.users
stats()
db.users
.find( {"cover_type":{$exists:true} },{book_name:
first_name:1, num_of_p=
.updateMany( { $and: [ { "Language": "Engl
"rating": {$lte: 4.3} } ] }, {$set: {"cover_type":"F
"is_popular":f=
.aggregate( [ {$match :
{"publication_year":{$gte:2
{"publication_year":{$lte:2018} } } ] }, {$group:
number of books":{"$sum:1} } }, {$project: { _id:C
number of'
.find( { $and:[ {"publication_year":{$gte:2
{"publication_year":{$lte:201
}
```

Neo4j Script to Create Nodes and Relati

Create City Nodes:

```
CREATE (nyc:City {name: 'New York'})
CREATE (la:City {name: 'Los Angeles'})
CREATE (chicago:City {name: 'Chicago'})
CREATE (dallas:City {name: 'Dallas'})
CREATE (cleveland:City {name: 'Cleveland'})
```

Create Flight Relationships with Distances and Additional Properties:

```
CREATE (nyc)-[:FLIGHT {distance: 2451, duration: 300, pilot:'Chris Johnson'}]->(chicago)
CREATE (nyc)-[:FLIGHT {distance: 790, duration: 120, pilot:'Jane Smith'}]->(la)
CREATE (la)-[:FLIGHT {distance: 1235, duration: 210, pilot:'Nancy Wilson'}]->(chicago)
```


1. CRUD Operations

- insertOne(document)

```
db.users.insertOne({ name: "Alice", age: 25, city: "New York" });
```

- insertMany([documents])

```
db.users.insertMany([
  { name: "Bob", age: 30, city: "San Francisco" },
  { name: "Charlie", age: 35, city: "Chicago" }
]);
```

- find(query, projection)

```
db.users.find({ city: "New York" }, { name: 1, age: 1 });
```

- findOne(query, projection)

```
db.users.findOne({ name: "Alice" }, { age: 1, city: 1 });
```

- updateOne(filter, update, options)

```
db.users.updateOne(
```

```
  { name: "Bob" },
```

```
  { $set: { age: 31 } }
```

```
);
```

- updateMany(filter, update, options)

```
db.users.updateMany(
```

```
  { city: "San Francisco" },
```

```
  { $set: { city: "Los Angeles" } }
```

```
);
```

- replaceOne(filter, replacement, options)

```
db.users.replaceOne(
```

```
  { name: "Charlie" },
```

```
  { name: "Charlie", age: 36, city: "Chicago", job: "Engineer" }
```

```
);
```

- deleteOne(filter)

```
db.users.deleteOne({ name: "Alice" });
```

- deleteMany(filter)

Brack

```
db.BrackCollection.find({$or:[{  
  "best-seller":{"exists:true"}},{  
  "book-type":{"History"}}]}},{_id:0,title:1,description:  
1},author_name:1})
```

(1.1)

```
db.BrackCollection.find({$and:[{  
  "publication_year":{"$gte:2023"}},{  
  "cover_type":["Paperback"]}]}).  
count()
```

3

(1.2)

```
db.BrackCollection.aggregate([  
  {  
    $match:{  
      $and:[{  
        "publication_year":  
        {"$gte:2023"}},{  
        "cover_type":  
        "Paperback"}]}]}],  
  {  
    $group:{  
      "Total  
number of books":  
      {$sum:1}},  
      {$project:  
      {_id:0,"Total number of books":1}}])
```

2

(1.3)

```
db.BrackCollection.updateMany(  
  {$and:[{"rating":10},{  
    "cost":{"$lte:50}}]}],  
  {$set:{  
    "cost":50}})
```

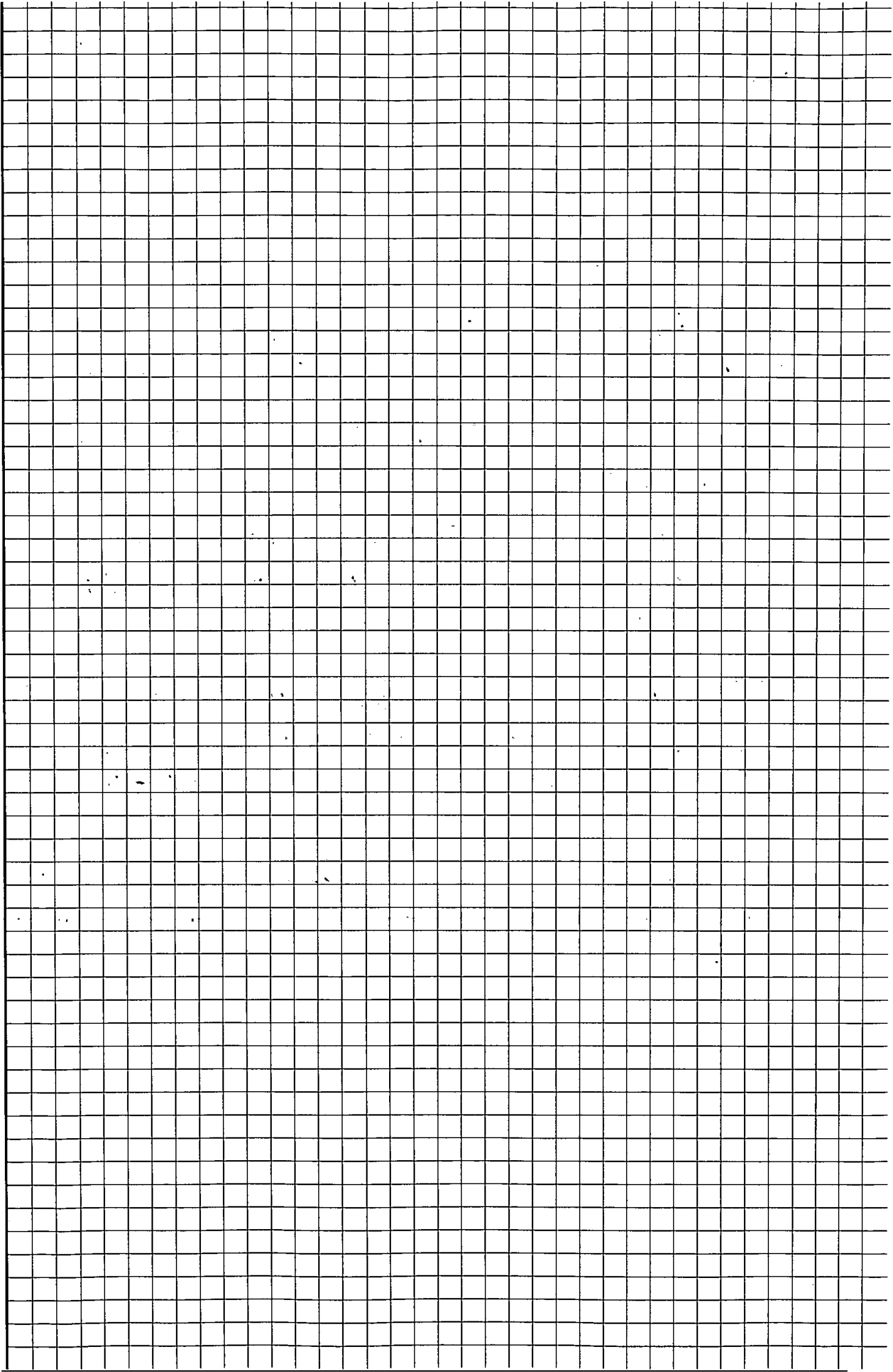
3

(1.4)



היה נדרש להשתמש באופרטור MAX

היה צריך להגדיר id = ונתון כל קבוצה לספור את האיברים שבה



5

(1.5)

-7

אנדי

db.BraaCollection.find(~~{~~sort

~~{~~publication_year

~~{~~find("publication_year": {"\$gt": 2013})

~~{~~sort({"cost": 1}).limit(3).skip(2)

צריך "לכבות" את id

(לא הורדתי ניקוד)

{ "title": "id:0", "book_type": 1 }

db.BraaCollection.aggregate([

{ \$match: { "publication_year": {

{ \$gt: 2013 } }],

{ \$sort: { "cost": 1 } },

צריך "לכבות" את id

(לא הורדתי ניקוד)

{ \$limit: 3 },

{ \$skip: 2 }

])

5

(1.0)

-15

0
(1.8)

0
(1.7)

db.BracketsCollection.aggregate()

~~{match:groupof matches}~~

{group:{id:"\$language",total:

{sum:"\$language"}}

\$sum:1

{sort:{total:-1}}

{limit:1}

נדרש היה גם להוסיף project

])

5

(19)

ניקוד התרגיל הזה הינו 7 נקודות מתוך 9.

הורדתי 2 נקודות נוספות על כך שברוב הסעיפים שלמעלה לא הקפדתי על מה שגם כתוב

בהוראות המבחן וגם דיברתי על זה רבות בכיתה: לא "לשבור שורות" באמצע "internal json"

לכן בסכ"ה ניקוד הסעיף הזה הינו 5 נקודות.

0
(1.10)

<Company>

<~~Brack~~Company/Name~~B~~ackCompany

<Manager/Name~~B~~ackHijack

<employees> managerName

<employeeName=BrackHijack

salary=1000 empAddress=115

emp/Name hijack BrackLast="Brack"

First="Brack"

<First>Brack/First>

<last>Hijack/last>

<empName>
~~<salary></salary>~~

<salaryAmount="1000"></amount>

<currency="5000">currency

</salary>

<empAddress>

<city></city>

<street></street>

<houseNum></houseNum>

</empAddress>

<CompanyID=71>

<empID=212266576>

<ManagerID ID=212266576>

<type FullTime>

<managerID "1" "3">

<employer>

<employee>

<empName>

<first> John <last>

<last> Hay <last> <empName>

<salary>

<amount> 1000 <amount>

<currency> \$ <currency>

</salary>

<empAddress>

<city> K </city>

<street> 3 </street>

<houseName> HouseName >

<empAddress>

<companyId ID="1">

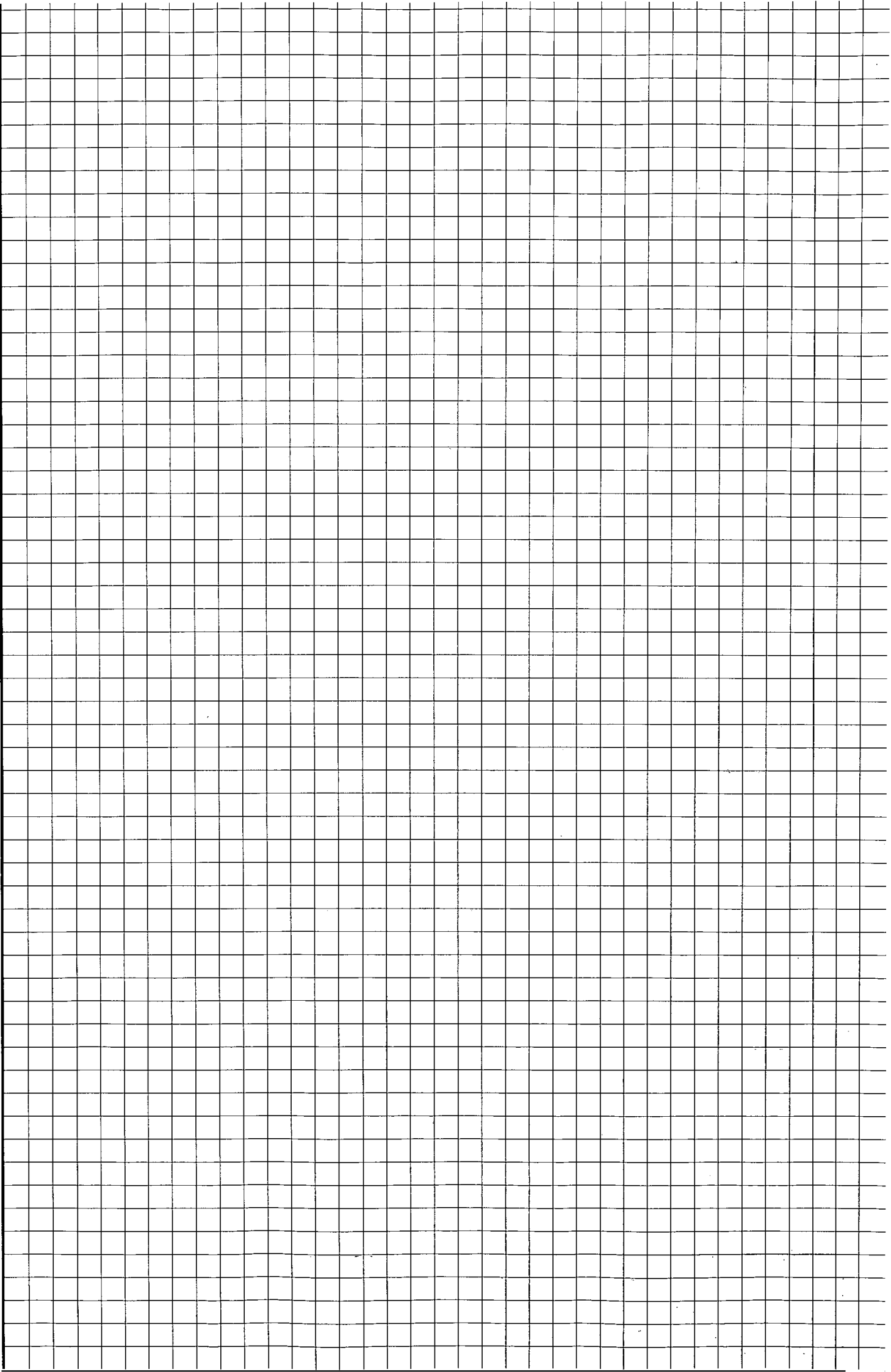
אלו לא אלמנטים

<empID ID="1">

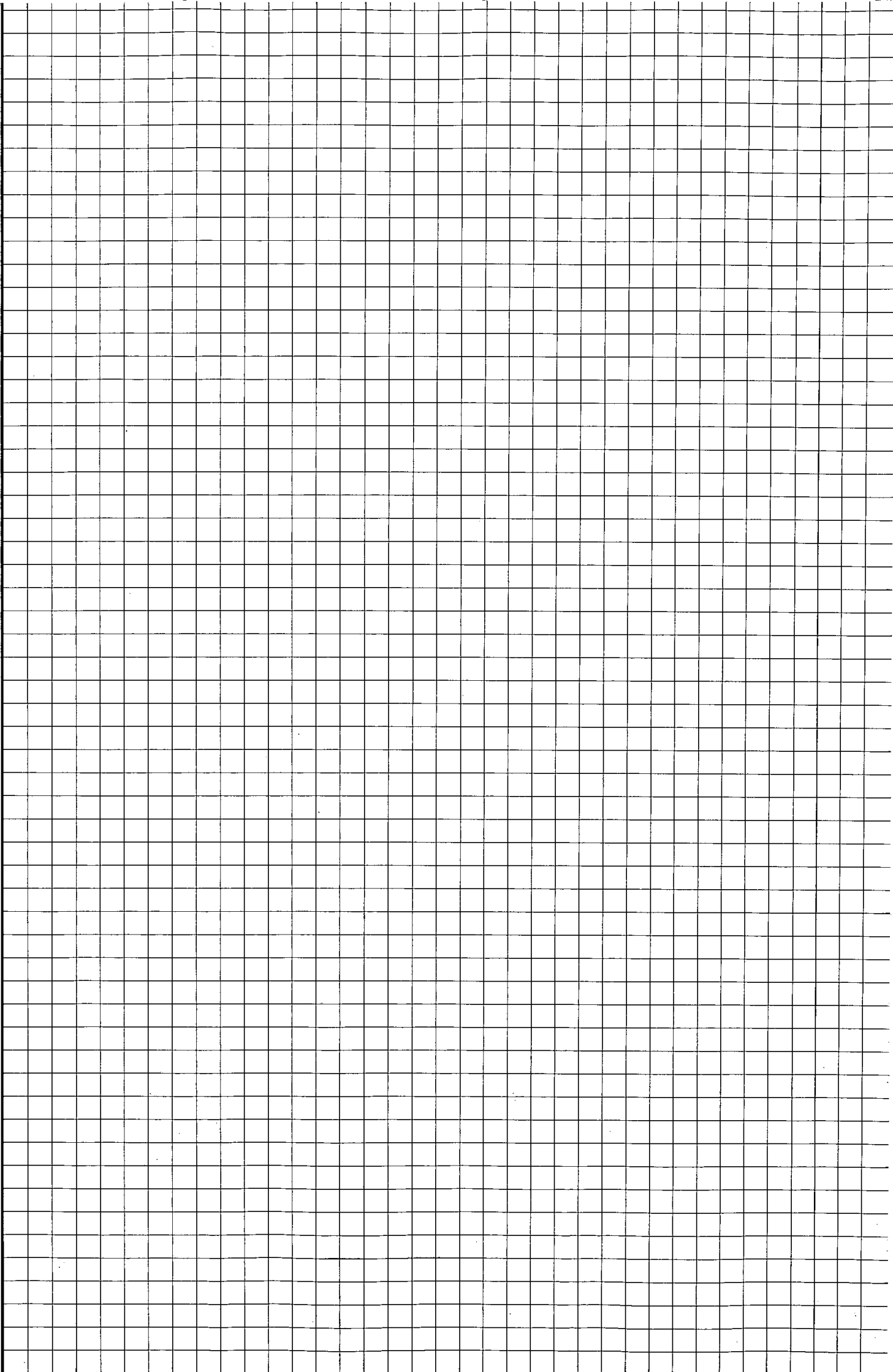
אלא תכונות.

<employee managerID=

<employee type="FullTime">



<employee managerId="2">
 <employee>
 <employee>
 <empName>
 <first>Son</first>
 <last>not</last> <empName>
 <salary>
 <amount>1000</amount>
 <currency>500</currency>
 </salary>
 <empAddress>
 <city>HK</city>
 <street>7st</street>
 <houseNum>7</houseNum>
 <empAddress>
 <company companyId="1">
 <employee empId="2">
 <employee managerId="1">
 <employee type="Student">
 <employee managerId="null">
 </employee>



לא כל התכונות ממומשות בהצלחה בקוד שכתבת.

יש אי דיוק במימוש של תכונות לעומת אלמנטים.

לא מומשה בהצלחה ההיררכיה של העובדים כפי שנדרשה בשאלה, ויש טעויות במימוש
managerOf.

לנבי הכתב- התייחסתי למעלה, היה ניתן לפחות לרווח מעט בין השורות...

הניקוד מאוד מתחשב..

15

(3)

0

(2.1)

0

(2.2)

0

(2.3)

0

(2.4)

