Homework 6

General Instructions

1. In this homework assignment you are going to write a series of Mongo DB queries. Place each query in a separate file named 1.js, 2.js, 3.js and so on where the number is the question number.
2. Make the queries be nicely formatted. For example in the find operations put the find and projection operations on different lines and in the aggregation operations place each aggregation operation on a different line. It also helps to put different fields on different lines.
3. Please make sure that your Mongodb queries compile and work properly since we will be grading them by running them in the mongodb shell.
4. The name of your database must be students and all your queries will start with the prefix "`db.students`".
5. You can use the mongo db shells provided at https://docs.mongodb.com/manual/tutorial to test your queries. You will need to go to a page that contains a mongo shell, such as this one.
6. Use the following schema for your queries:

```
students(lastname, firstname, dateOfBirth, email, grade, major, [quizzes],
  report { score, ta, comments }, exams [ { score, ta, comments } ] )
```

   Here is a sample document:

```
{ lastname: "mouse", firstname: "mickey",
  dateOfBirth: new Date("2000-05-15"), email: "mickey@disney.com",
  grade: "B+",
  major: "CS",
  quizzes : [85, 76, 91, 100, 64, 91],
  report : { score: 78, ta: "Jeff Hardy", comments: "too many mice" },
  exams : [ { score: 92, ta: "Nancy Hew", comments: "nice code!" },
            { score: 87, ta: "Nancy Hew", comments: "nice style" },
            { score: 91, ta: "Nancy Hew" } ]
```

   You can find more sample data in /home/bvanderz/cs465/hw/hw6/mongo_setup.js. The first question has you insert a new document into the database. You may also want to add some sample data of your own.
7. You may assume that the email field provides a unique identifier for each document.
8. I have provided sample output for some but not all of the questions. You should be able to figure out what I expect from the sample output that I have provided. MongoDB does not allow you to do any pretty printing so other than getting the order of the fields correctly printed, do not worry about trying to pretty print the output.

Write a Mongodb query to implement each of the following operations and please observe the following points:

- **On all the queries that ask you to print something, exclude the _id field in the printed results. The one exception is on any aggregation question that requires the group operator. For those questions you should print the _id field**.

- When you indicate in your queries which fields to project, please list the fields in the order I list them in the problem. However, you will notice that the order in which you list them in the projection filter may not be the order that they appear in the output. That's fine. MongoDB prints fields in the order they appear in the document, not in the order they appear in the projection filter.

1. Insert a new document with the following information and put the fields in the exact same order as I placed them here:
    - lastname: duck
    - firstname: donald
    - date of birth: 1993-03-17
    - email: dduck@disney.com
    - grade: C
    - major: MATH
    - quizzes: 43, 71, 53, 100, 0, 32, 31, 41
    - report: score: 71, ta: Clark Richards
    - exams
        1. score: 58, ta: "Nancy Hew", comments: "Need to step it up"
        2. score: 69, ta: "Clark Richards", comments: "Still not passing"
        3. score: 45, ta: "Nancy Hew"
        4. score: 33, ta: "Nancy Hew"
        5. score: 61, ta: "Clark Richards", comments: "Stop laying eggs on these exams"

    To help the TA verify your answer, follow the insert with the following query:

    ```
    db.students.find({ email : "dduck@disney.com" }, { _id : 0 })
    ```

2. Print the first and last names of all students who received an A in the course. Sample output might be:

    ```
    { "lastname" : "vander zanden", "firstname" : "smiley" }
    { "lastname" : "mouse", "firstname" : "minnie" }
    ```

3. Print the first name, last name, date of birth, and grade of all students born in the year 2000. Sample output might be:

    ```
    { "lastname" : "mouse", "firstname" : "mickey", "dateOfBirth" : ISODate("2000-05-15T00:00:00Z"), "grade" : "B+" }
    { "lastname" : "sublime", "dateOfBirth" : ISODate("2000-01-18T00:00:00Z"), "grade" : "C" }
    { "lastname" : "ape", "firstname" : "great", "dateOfBirth" : ISODate("2000-01-18T00:00:00Z"), "grade" : "B" }
    ```

    Notice that the dates are ISODate objects with timestamps starting at midnight. That is fine--the ISODate objects with timestamps are created by the "new Date()" command.

4. Print the first name, last name, and quizzes of all students who received an A and who had at least one quiz with a score of less than 50. Sample output might be:

    ```
    { "lastname" : "mouse", "firstname" : "minnie", "quizzes" : [ 100, 94, 98, 35, 97, 100, 100 ] }
    ```

5. Print the first name, last name, and exams of all students who had at least one exam graded by Jeff Hardy. Print the results in alphabetical order by lastname and if there are multiple documents with the same last name, then in reverse alphabetical order by firstname (i.e, first names would go from z down to a). Sample output might be:

```
{ "lastname" : "mouse", "firstname" : "minnie",
"exams" : [ { "score" : 83, "ta" : "Jeff Hardy", "comments" : "problem 1 was tough" },
            { "score" : 97, "ta" : "Clark Richards", "comments" : "excellent!" },
            ... ]}
{ "lastname" : "mouse", "firstname" : "mickey",
  "exams" : [ { "score" : 92, "ta" : "Nancy Hew", "comments" : "nice code!" },
              { "score" : 87, "ta" : "Nancy Hew", "comments" : "nice style" },
            ... ]}
```

   Note that this is abbreviated output and I have used "..." so I don't have to show all of the exams (your output should show all of the exams for each document). If you use my sample data there will be more output for this query. I showed the two mouses to show how the sorting should be done. I have also used line-breaks to fit this output into the width of a browser window. In your actual output it will be one long line of output.

6. Print the first name, last name, and the report score of any student who had a report score between 60 and 80 inclusive and was graded by Nancy Hew or who had an exam score between 80 and 100 inclusive and was graded by Nancy Hew.

7. Print the first and last names and reports of all students who did not receive a comment on their report.

   For the following aggregation questions, I have put in parentheses the name you should give to the aggregated field. For example, if I say print the avg (avgQuiz) of a student, I want the output field to be named avgQuiz.

8. Print a count (count) of the number of students by major. Sample output might be:

```
{ "_id" : "CS", "count" : 4 }
{ "_id" : "MATH", "count" : 4 }
{ "_id" : "CPE", "count" : 1 }
```

   You have to use "_id" to group so the query will have to output the `_id` attribute.

9. Print the first name, last name, and the minimum (min), maximum (max), and average quiz score (avgQuiz) of each student.

10. Print the first name, last name, and average exam score (avgExam) of each student. Print the students in descending order by average exam score.

11. Print a count (count) of the number of students who received a C, D, or F in the course. For example, if 3 students received a C, 5 students received a D, and 2 students received an F, then your count would be 10. Sample output for this query might be:

```
{ "count" : 10 }
```

For the following document modification queries I have you write the modification query and then a second query that allows the TA to verify that your modification query worked.

12. Update Mickey Mouse's grade to an A and change the TA who graded Micky Mouse's report to Clark Richards. Then print Micky Mouse's document, excluding the _id field. Sample output might be:

```
{ "lastname" : "mouse", "firstname" : "mickey", "dateOfBirth" : ISODate("2000-05-15T00:00:00Z"),
  "email" : "mickey@disney.com", "grade" : "A", "major" : "CS",
  "quizzes" : [ 85, 76, 91, 100, 64, 91 ],
  "report" : { "score" : 78, "ta" : "Clark Richards", "comments" : "too many mice" },
  "exams" : [ { "score" : 92, "ta" : "Nancy Hew", "comments" : "nice code!" },
              { "score" : 87, "ta" : "Nancy Hew", "comments" : "nice style" },
              { "score" : 91, "ta" : "Nancy Hew" } ] }
```

I have used line-breaks so the output fits into the width of a browser window. Your output will be one long line.

13. Add an exam to Smiley Vander Zanden with a score of 95 and a TA with the name "Jeff Hardy". Then print Smiley Vander Zanden's document, excluding the _id field.

14. Update Minnie Mouse's first exam score to be an 86 and her third exam score to be 100 and the ta to be Nancy Hew. Then print Minnie Mouse's document, excluding the _id field. In the sample data that I provided, Minnie's first exam score was a 83 and her third exam score was a 90.

15. Delete great ape's document. Then write a query that tries to find great ape.