DS 503 - Project 5
By :
Abdulaziz Alajaji - asalajaji@wpi.edu
Yousef Fadila - yousef@fadila.net

Q1)

Parent references - add the objects

```
db.categories.insert( { _id: "MongoDB", parent: "Databases" } )
db.categories.insert( { _id: "dbm", parent: "Databases" } )
db.categories.insert( { _id: "Databases", parent: "Programming" } )
db.categories.insert( { _id: "Languages", parent: "Programming" } )
db.categories.insert( { _id: "Programming", parent: "Books" } )
db.categories.insert( { _id: "Books", parent: null } )
```

1)

```
var results=[];
var parentId = db.categories.findOne({_id: "MongoDB"}).parent;
var level = 1;
while(parentId){
   parent = db.categories.findOne({_id: parentId});
         result = {name: parent._id, level: level};
         results.push(result);
         level = level + 1;
         parentId = parent.parent;
}
results
```

2)

```
function heightOfTree(r_nodeId)
{
   var max = 0;
   var children =  db.categories.find({parent: r_nodeId});
   while (children.hasNext() == true) {
      var child = children.next();
      child_height = heightOfTree(child._id);
      if (child_height > max)
            max = child_height;
     }
   // it will return 1 (1+max:0) in case no children(recursive stop condition)
   return 1 + max;
}

var root = "Books";
print(heightOfTree(root));
```

Children references - add the objects

```
db.categories.insert( { _id: "MongoDB", children: [] } )
db.categories.insert( { _id: "dbm", children: [] } )
db.categories.insert( { _id: "Databases", children: [ "MongoDB", "dbm" ] } )
db.categories.insert( { _id: "Languages", children: [] } )
db.categories.insert( { _id: "Programming", children: [ "Databases", "Languages" ] } )
db.categories.insert( { _id: "Books", children: [ "Programming" ] } )
```

---

3)

```
var parent = db.categories.findOne({children: "dbm"});
print(parent._id);
```

4)

```
var descendants = [];
var stack = [];
var item = db.categories.findOne({_id: "Books"});
stack.push(item);
while (stack.length > 0) {
    var current = stack.pop();
    var children =  db.categories.find({_id: {$in: current.children}});
    while (children.hasNext() == true) {
            var child = children.next();
            descendants.push(child._id);
            stack.push(child);
            }
}
descendants;
```

5)
It is not clear from the question if we need to report **ids only** or the **whole objects.**
Please find the solutions of both versions below

```
// Report siblings objects.
var parent = db.categories.findOne({children: "Databases"});
var siblings =  db.categories.find({$and : [{_id: {$in: parent.children}}, {_id: { $ne :"Databases"}}] });
siblings

// Report siblings ID only

var parent = db.categories.findOne({children: "Databases"});
var siblings = [];
for (var i = 0, len = parent.children.length; i < len; i++) {
    if (parent.children[i]!= "Databases")
            siblings.push(parent.children[i]);
}
siblings
```

Q2)

1)

```
db.test.mapReduce(
   function() {
          if (typeof this.awards != "undefined") {
              this.awards.forEach(function (award) {
              emit ( award.award , 1) ;
          });
          }

   } ,  function(key, values) {
      return Array.sum( values);
   }, {
       out : 'q1_out'
   });
```

2)

```
// record 10 doesn’t have birth defined, we created a special group (using $cond) to group
// such records without birth; group id would be records without $birth

db.test.aggregate([{
       $group:
        {
           _id : { year : {$cond: [{ $ifNull: ['$birth', 0] }, {$year : "$birth" }, "records without $birth"] }},
           records_ids : { $addToSet: "$_id" }
        }
}]);
```

3)

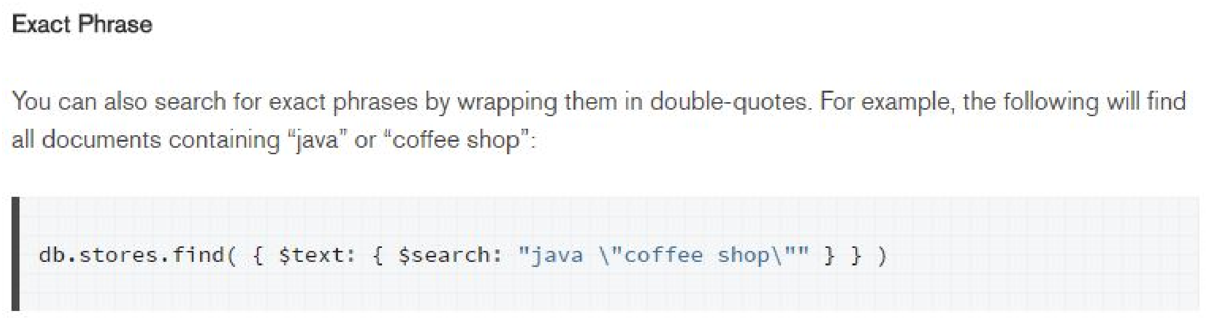
```
var myCursor = db.test.aggregate([
      {$group : {_id : null,
          min_id: { $min: "$_id" },
          max_id: { $max: "$_id" }
       }}]);
var min_max = myCursor.next();

var min_id_object =  db.test.find({_id : min_max.min_id});
min_id_object

var max_id_object =  db.test.find({_id : min_max.max_id});
max_id_object
```

4)
db.test.createIndex({"$**": "text"});
db.test.find({ $text: { $search: "\"Turing Award\"" } });

5)
According to https://docs.mongodb.com/v3.2/text-search/



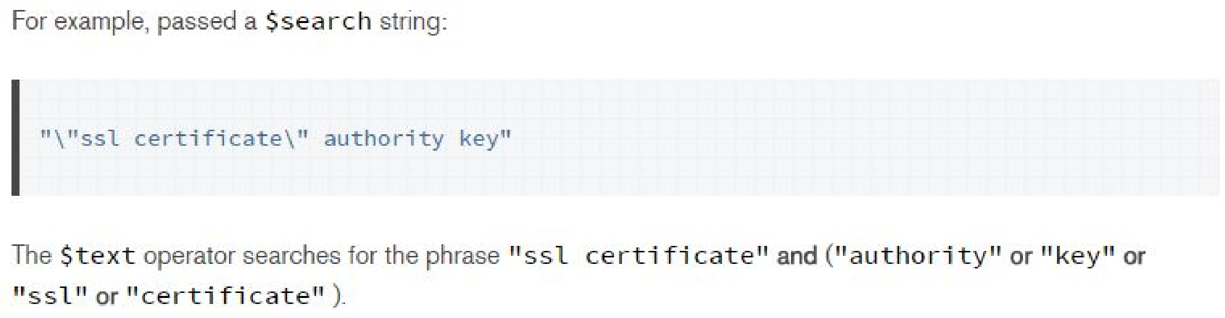
Exact Phrase

You can also search for exact phrases by wrapping them in double-quotes. For example, the following will find all documents containing “java” or “coffee shop”:

```
db.stores.find( { $text: { $search: "java \"coffee shop\"" } } )
```

This query should give the result.
db.test.find({ $text: { $search: "Turing \"National Medal\"" } })

But it didn’t work as it seems there is an error in the documentation or it is maybe related to version. In our tries, the behaviour was And, and it is actually described on:
https://docs.mongodb.com/manual/reference/operator/query/text/#text-query-operator-behavior



For example, passed a `$search` string:

```
"\"ssl certificate\" authority key"
```

The `$text` operator searches for the phrase `"ssl certificate"` and (`"authority"` or `"key"` or `"ssl"` or `"certificate"`).

So as
db.test.find({ $text: { $search: "Turing \"National Medal\"" } })
Didn’t work in 3.4 version and a query can specify, **at most**, one $text expression. **we did that in 2 queries and return the union of them.**

```
function isExist(arr, itm) {
    for (i = 0; i <arr.length; i++ ) {
        if (arr[i]._id == itm._id )
            return true;
        if ((typeof itm._id.str != "undefined") && arr[i]._id.str == itm._id.str)
            return true;
    }
    return false;
}

var result1 = db.test.find({$text:{$search: "Turing"}});
var result2 = db.test.find({$text: {$search: "\"National Medal\""}});
var finalResult = [];

result1.forEach(function(record){
  finalResult.push(record);
})

result2.forEach(function(record){
if (isExist(finalResult, record) == false)
    finalResult.push(record);
})

finalResult;
```