Informatics 1: Data & Analysis
Session 2012–2013

Exam Feedback

This is a feedback report on the end-of-year examination for the course *Informatics 1: Data & Analysis*, held on Monday 29 April 2013. It reviews the exam itself and gives feedback on the solutions submitted by the 195 students who completed the course. Please note the following:

- This is not a set of "model" answers. It does contain solutions, which can be used to check your own answers; but there are also extended discussions of different possible answers, key points, possible errors, and comments on the ways people approached each question on the day.
- Not all the questions have a single "right" answer. There can be multiple correct ways to write a database query, explain a concept, or construct an example. This report includes several variants on answers, but still cannot cover every possible correct alternative.
- Studying past papers is one way to learn more about a subject, but it is quite limited and not enough on its own. Even when an exams routinely follow a fixed structure, the questions change and successful performance does essentially depend on a good understanding of the material in the course.

The exam consisted of three questions, each with several subquestions. The rest of this report is a one-page summary and then the full text of each question followed by notes on solution and feedback on the answers given.

Where you find errors in these notes, please send them to me at Ian.Stark@ed.ac.uk

Ian Stark
2013-07-11

Summary

Overall grades in this exam were very high, with a large number of excellent answers to individual questions. For every part of the paper, there were examples of crisp, clear, and correct solutions, well set out and easy to understand.

These didn't always come together, though, and many people mixed good answers in one area with partial or uncertain attempts in others — there was no part which every candidate got correct.

There were a lot of cases where people gave good answers to technical calculations or formal coding, but matched with much weaker written explanations of what it was they had done. This was a common theme, and for anyone looking to improve their performance I strongly recommend practice on parts of questions requiring explanation and definitions; and not just thinking about whether you know what's happening, but actually writing out a complete solution then reading it back, looking up the material, and trying to improve what you have written.

Question 1. There were lots of good entity-relationship diagrams here, although some people put in extra arrows and double lines which didn't match the requirements given. SQL queries were done well, too. Weak points were the standard definitions, and more generally not being able to give clear and precise explanations of what was happening.

It's important not only to be able to do things (code SQL, draw ER diagrams) but also to write about them and effectively communicate your understanding. Practice can help here: attempt an explanation unaided, then try improving it after looking at lecture slides, your own notes, a textbook, or online resources.

Question 2. All sorts of trees drawn here, many getting things mostly right but not all pinning down details like the root node and the correct use of attributes. There were lots of good XPath queries, correctly answering the questions in various different ways.

Several people did have difficulty with the DTD part: not just the syntax and format, but particularly the use of regular expressions to describe element structures. Regular expressions, with "|", "+", "*" and so on, appear in all kinds of computing applications. This makes them well worth learning about; and also means that there are lots of descriptions and resources online to help.

Question 3. Again, many good calculations but not as many clear explanations of what was being computed. If you are preparing by working on a past exam question, try writing out your whole answer and then reading it some time later to see how clearly you can understand it yourself.

In ranking documents by cosine measure, quite a few people compared the documents to each other, which is wrong — you need to compare each document individually against the original query, to see which is the best match.

Question 1 [*This question is worth a total of 35 marks.*]

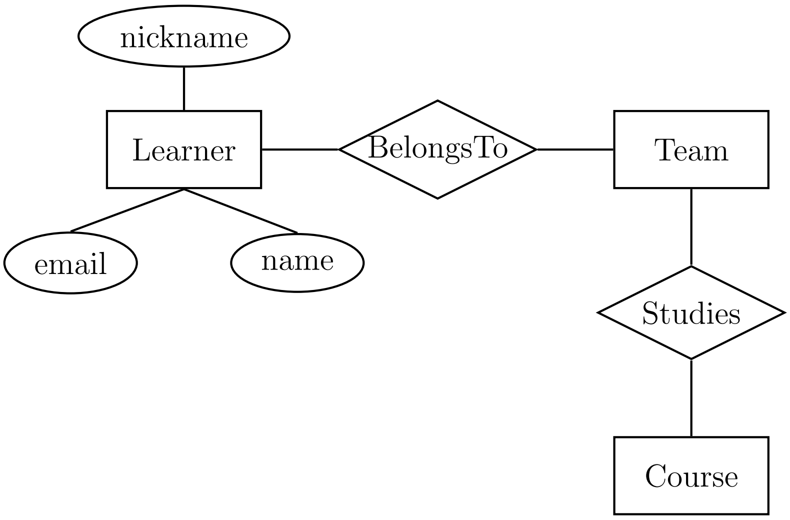
The non-existent startup company *W!L!T!* plans to create a collaborative online community of learners. From their breathless sales pitch:

> Come and join with *We! Learn! Together!* where everyone helps each other! Join teams of other learners just like you, and work together on a host of online study courses! Earn star points as you help others to learn too! You're never alone as We! Learn! Together!

Their plans for a website include a requirement to keep track of the following data.

- For every learner their name, unique identifying email address, and an optional nickname.
- For every study team its title and unique 8-character alphanumeric ID.
- Which learners belong to which teams, where anyone can belong to any number of teams.
- All courses, their titles and the number of star points a team can gain by completing the course.
- Details of which team is studying which courses.

The picture below shows an incomplete entity-relationship (ER) diagram representing some of this information.



(a) Give examples of an *entity set*, a *relationship* set, and an *attribute* from this diagram. [*3 marks*]

(b) Give brief explanations of the following terms: *key*, *candidate key*, *primary key*, *composite key*. [*8 marks*]

(c) Copy and complete the ER diagram, including any missing attributes and choosing appropriate keys. [*7 marks*]

(d) The organisers decide that some learners are more equal than others and so allow study teams to nominate one of their members as leader, if they wish. Extend the ER diagram with a relationship recording this new information. How do you express the constraint that each team can have only one leader? [*6 marks*]

(e) Elsewhere the *W!L!T!* database has the following SQL for tables describing quizzes in the online courses, where each quiz is made up of several questions.

```
create table Quiz (
    id      varchar(8),
     title  varchar(30) not null,
    primary key (id)
)

create table Question (
    id      varchar(8),
    marks   integer not null,
    quiz    varchar(8) not null,
    primary key (id),
    foreign key (quiz) references Quiz(id)
)
```

What is the effect of the constraint in the last line, beginning **foreign key** ...? [*2 marks*]

(f) Write an SQL query to count the number of questions held in the database. [*3 marks*]

(g) Write an SQL query to find every 10-mark question, listing the title of the quiz it appears in and the question id. [*6 marks*]

Notes on Question 1

(a) The entity sets in this diagram so far are *Learner*, *Team* and *Course*. The relationship sets are *BelongsTo* and *Studies*. The attributes are *nickname*, *email* and *name*.

Note that the question only requires one of each of these. This was far and away the most correctly answered item in the exam. Well done, folks.

(b) **Key** A *key* is a minimal set of attributes whose values uniquely identify each entity in an entity set.

Candidate key Where there is more than one such set, each one is a *candidate key*.

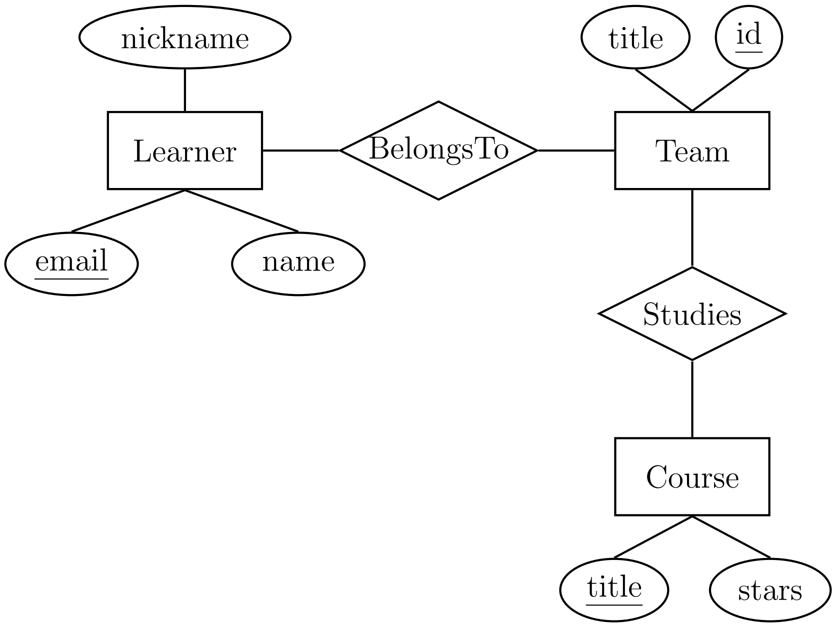
Primary key Where there are several different candidate keys, one is chosen as the *primary key*.

Composite key If a key comprises more than one attribute, then it is a *composite key*.

This is "bookwork", in that these are standard definitions, but it's still easier to get a good answer if you understand what's going on. Quite a few people missed out some parts, or got these the wrong way round. Here are some important points.

- In general a key is a *set* of attributes, not just a single attribute. In fact, even when only one attribute is required to uniquely identify each entity, the associated key is still the one-element set containing that attribute. When it takes more than one attribute to uniquely identify each entity, they make up a composite key.
- Some answers mixed up composite keys with weak entities. These aren't necessarily connected — any kind of entity set may have a composite key.
- The set of attributes making up a key has to be *minimal*, meaning that none of the attributes can be left out. Here, for example, the set of attributes {*email*, *name*} will uniquely identify each *Learner*, but that's not a key as we can safely just use {*email*}. (In fact, the larger attribute set is called a *superkey*, but that name wasn't mentioned in the course this year.)

(c) Here is a complete diagram.

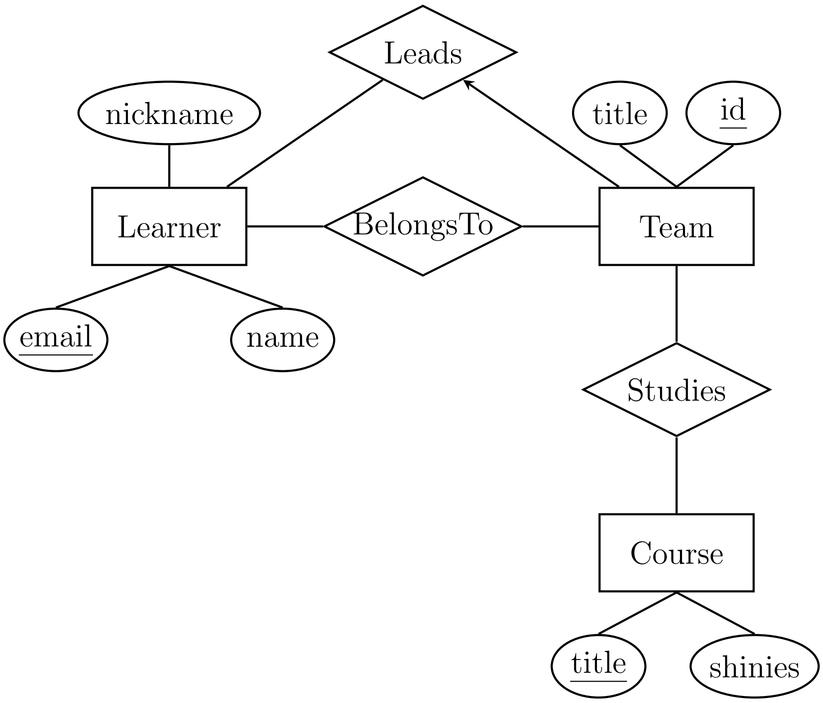


The additional items are underlining of the *email* attribute to show it forms part of the key (in fact, the entire key); and all the attributes on *Team* and *Course* entities, with their key sets of attributes {*id*} and {*title*} underlined.

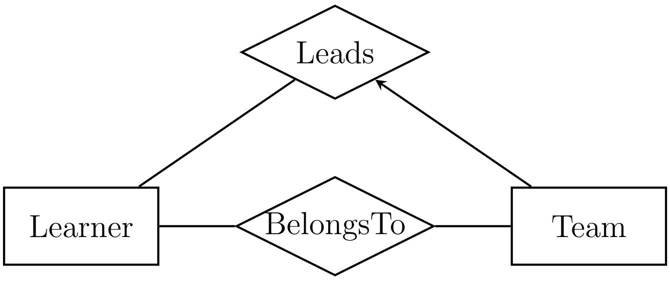
There is no need to add any thick or double lines (for participation constraints) or arrowheads (for key constraints). Those would only be necessary if the requirements stated that, for example, every learner must belong to at least one team, or that every course can only be studied by one team.

Although in general it is possible to have attributes on relationships like *Studies* and *BelongsTo*, there's no need to do that here.

(d) This revised ER diagram adds a *Leads* relationship to show where someone is leader of a team. There is a *key constraint*, represented by an arrow on the line joining *Team* to *Leads* to indicate that each team can participate in at most one such relationship; i.e. each team has at most one leader.



It's important in this question to indicate which part is new in the diagram. Here I've written a text explanation, but some people showed this by adding the new elements in colour to their original diagram, with a note explaining which parts answered which question. Another possibility is to draw only the additional elements, like this:



The extended ER diagram has an additional *Leads* relationship between *Learner* and *Team*.

One thing that doesn't work too well is just adding the relationship to an existing diagram, without indicating which parts are new. Some people did that, which made it impossible to tell which parts of the diagram are answering which part of the question.

Considering various possible situations allows us to explore whether there should be any other annotations on the diagram. It's possible for someone to be leader of more than one team (so no arrow on the line from *Learner* to *Leads*); for someone to be the leader of no team (so no thick or double line of *total participation* from *Learner* to *Leads*); and for a team to have no leader (so no thick or double line from *Team* to *Leads* either).

Some people did make those additions, which was a mistake. Also, a few were reluctant to set up a situation where two different relationships (*Leads* and *BelongsTo*) link the same entity sets (*Learner* and *Team*). They didn't need to worry: in complex databases entities are often found in many different kinds of relationship at the same time.

It is possible to introduce a *Leader* subclass of entity to *Learner*, but while correct it doesn't particularly help as leaders don't have any additional attributes. However, in this case the *Leads* relationship should join *Leader* and *Team*, and should also have a thick or double line from *Leader* to *Leads* because every leader needs at least one team to lead.

One incorrect approach, which a few people tried, is to add an additional attribute *leader* to *Team*. This doesn't satisfy the question asked, which specifically requires a new relationship. In practice it also gives no assurance that the named leader is a *Learner*, and even if they are, it provides no way to recognise that connection.

(e) The effect of the constraint "**foreign key** (quiz) **references** Quiz(id)" is that any value in the quiz field of a Question table entry must always be the id of a row already in the Quiz table.

Note that the earlier line "quiz varchar(8) **not null**" ensures that every Question entry does in fact give some value to the quiz field.

(f) The following will count the number of questions held in the database.

```
select count(id) from Question
```

Various alternatives like **count(distinct** id) or **count**(∗) are fine too. The **distinct** keyword isn't essential, though, as all id values are unique in this table.

(g) This SQL query lists every 10-mark question, giving the title of the quiz it appears in and the question id.

```
select Quiz.title, Question.id
from Quiz, Question
where Question.quiz = Quiz.id and Question.marks = 10
```

This takes every combination of a row in the Quiz table and a row in the Question table where the quiz and the question match up, and the question scores 10 marks.

It's essential here to include the test "Question.quiz = Quiz.id" — several people missed this out, but without it the results returned won't match up the Question and Quiz correctly.

Some people gave slight variants that were still correct: for example, adding aliases for the table names, or even leaving out some of the "Question." and "Quiz." qualifiers altogether. For example, here is one alternative answer:

```
select title, Q.id
from Question as Q, Quiz as Z
where quiz = Z.id and marks = 10
```

This is correct, although I find it slightly harder to decipher. Not all qualifiers can be omitted, though, as the id field appears in both tables so is potentially ambiguous.

Question 2 [*This question is worth a total of 35 marks.*]

The following small XML document is a marked-up version of a speech from one of Shakespeare's plays.

```
<speech speaker=”First Witch”>
    <line>
        <w>When</w>
        <w>shall</w>
        <w>we</w>
        <w>three</w>
        <w>meet</w>
        <w>again</w>
    </line>
    <line>
        <w>In</w>
        <w>thunder</w>
        <punct>,</punct>
        <w>lightning</w>
        <punct>,</punct>
        <w>or</w>
        <w>in</w>
        <w>rain</w>
        <punct>?</punct>
        </line>
</speech>
```

(a) Draw this XML document as a tree, following the XPath data model. [*9 marks*]

(b) Write an XML DTD for a Speech document type to validate such speeches. Assume that every speech must have an identified speaker. [*12 marks*]

(c) Suppose a large XML document contains many such speeches, nested at various levels inside Plays, Acts, Scenes and so forth. Write XPath expressions to identify:

(i) All lines spoken by Macbeth

(ii) All speakers using the word "blood" in a speech.

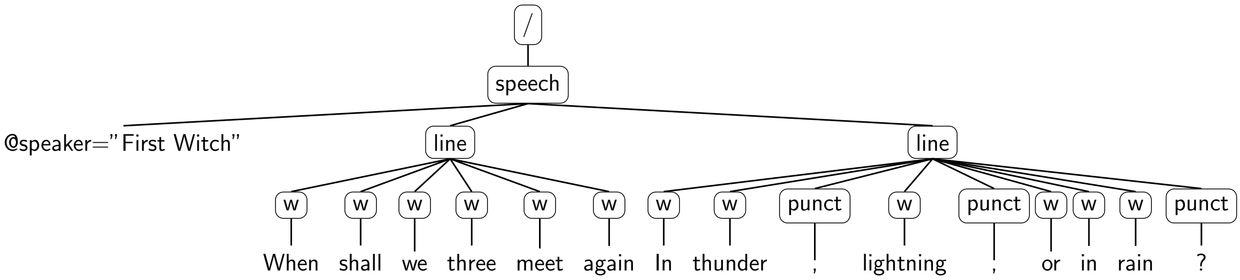
[*8 marks*]

(d) The lines above come from the works of a single author. Standard corpora for linguistic research like the *British National Corpus* or the *Penn Treebank* bring together work from many sources. Building them requires balancing and sampling in order to ensure that they are representative.

Explain the meaning of *balancing*, *sampling* and *representative* here. [*6 marks*]

Notes on Question 2

(a) The document has the following tree structure in the XPath data model.



Some things to note here: the root node "/"; the attribute node "@speaker=..."; both word and punctuation nodes with text at the leaves.

Some people used ellipses "..." to indicate further speeches and lines in the document. That's not correct here, as the question asked for the XPath tree of the document given.

(b) Here is an appropriate DTD.

```
<!ELEMENT speech (line)+>
<!ELEMENT line (w|punct)+>
<!ELEMENT w (#PCDATA)>
<!ELEMENT punct (#PCDATA)>
<!ATTLIST speech speaker CDATA #REQUIRED>
```

It would also be acceptable to add an enclosing <!**DOCTYPE** speech [...]> declaration, necessary if the DTD is to be embedded within the XML document itself.

It's important that the !ATTLIST declaration mentions both speaker (the attribute) and speech (the element to which it is attached). The question also says that every speech must have a speaker, hence the #REQUIRED tag.

Some people used different, correct, variations on the regular expressions for speech and line. It's possible, for example, to use "∗" for repetition instead of "+", although that then allows empty speeches and lines.

The following two incorrect attempts were quite common:

```
<!ELEMENT line (w+|punct+)>
<!ELEMENT line (w∗,punct∗)>
```

The first of these will only match lines that contain all words, or all punctuation, but not a mixture. The second will only match lines where all punctuation is at the end. Neither is general enough to capture the example document at the start of the question.

(c) (i) This XPath expression selects all lines spoken by Macbeth:

```
//speech[@speaker='Macbeth']/line
```

(ii) This identifies all speakers who use the word "blood" somewhere in a speech:

```
//speech[line/w/text()='blood']/@speaker
```

Here the excursion [line/w/text()='blood'] picks out speeches where the word "blood" appears, and then /@speaker returns the name of the speaker.

There are several possible variations on this, making more use of the descendant axis "//" or navigating upwards to parents with "..". For example:

```
//speech[.//w='blood']/@speaker
//*[w='blood']/../@speaker
```

Both of these correctly answer the question.

(d) **Balancing** means choosing a range of different types of sources for the corpus: books, newspapers, blogs, letters, etc.

Sampling refers to selecting texts at random from the chosen sources.

Representative A corpus is *representative* if it contains a similar mix of text to the language variant for which it is being developed.

Like question 1(b), this is “bookwork”, in that these are standard definitions; many people found it quite tricky to identify the different individual aspect of corpus building. Here you need not only a general idea about assembling a corpus, but also that a precise expression of the important features.

In this setting, *sampling* doesn’t refer to statistical tests, but to the random selection when picking some texts for the corpus instead of others. None of these terms were about analysing grammar or syntax, or tagging text with parts of speech.

Here are some alternative ways to express what it means for a corpus to be *representative*:

- It can be used to analyse the language it is chosen to represent;
- It can be used to test hypotheses about the language it represents.

Question 3 [*This question is worth a total of 30 marks.*]

(a) An information retrieval system is searching a European Parliament archive for documents on the topic of "offshore fishing boundary disputes". The following document matrix indicate three possible matches.

	offshore	fishing	boundary	disputes
Document A	4	2	7	0
Document B	3	3	3	3
Document C	12	6	0	0
Query	1	1	1	1

One way to rank these documents for potential relevance to the topic is the *cosine similarity measure.*

Write out the formula for calculating the cosine of the angle between two four-dimensional vectors (x_1, x_2, x_3, x_4) and (y_1, y_2, y_3, y_4).

Use this to rank the three documents in order of relevance to the query. [*10 marks*]

(b) One way to evaluate the performance of an information retrieval system is to assess its *precision* P and *recall* R. Informally, P can be defined as the proportion of the documents returned by the system which do match the objectives of the original search. Give a similar informal definition of R.

Here is the mathematical formula for calculating precision.

$$P = \frac{TP}{TP + FP}$$

Name and define the terms TP and FP here. Give the formula for recall R, explaining any other new terms that appear. [*8 marks*]

(c) You have been given two different information retrieval systems to compare: *Hare* and *Tortoise.* Each one is tested on the same query for a collection of 4000 documents, of which 200 are relevant to the query. *Hare* returns 1200 documents, including 150 that are relevant; while *Tortoise* returns just 160, with 120 of them being relevant.

Tabulate the results for each system and calculate their precision and recall on this test. Show your working.

One way to combine precision and recall scores is to use their *harmonic mean.* Give the formula for this, and calculate its value for each of *Hare* and *Tortoise.* [*12 marks*]

Notes on Question 3

(a) The cosine formula for four-vectors is:

$$\cos(\vec{x}, \vec{y}) = \frac{x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4}{\sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}\sqrt{y_1^2 + y_2^2 + y_3^2 + y_4^2}}$$

This explicitly uses the coordinate values provided in the question. There is an alternative presentation using only vector notation:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \,.\, \vec{y}}{|\vec{x}||\vec{y}|}$$

To rank the three documents, use this formula to calculate the cosine between each document and the original query.

$$\begin{aligned}
\cos(\text{Document A}, \text{Query}) &= \frac{4 + 2 + 7}{\sqrt{4}\sqrt{4^2 + 2^2 + 7^2}} &= 0.78 \\
\cos(\text{Document B}, \text{Query}) &= \frac{3 + 3 + 3 + 3}{\sqrt{4}\sqrt{3^2 + 3^2 + 3^2 + 3^2}} &= 1 \\
\cos(\text{Document C}, \text{Query}) &= \frac{12 + 6 + 0}{\sqrt{4}\sqrt{12^2 + 6^2}} &= 0.67
\end{aligned}$$

This ranks the three documents in order as:

- Document B
- Document A
- Document C

The largest cosine value indicates the document most closely matching the query.

While it's possible to go on and compute the angle between the each document vector and the query, it's not necessary. For cosine similarity, you only need to rank the cosine values in order.

Several people instead calculated the cosine between pairs of documents: A and B, B and C, C and A. That's incorrect, and definitely doesn't help rank them by relevance to the query: the "similarity" part of cosine similarity is between each document and the query.

(b) The *recall* R is the proportion of relevant documents in the collection which are successfully retrieved.

As with "balancing" and "sampling" in Question 2, it's important here to be able to precisely express what a technical term like "recall" signifies, as well as having a general understanding of the context. For example, notice that both precision and recall are a *proportion* of documents (which will be a real number between 0 and 1), not the total number (which will be some non-negative integer).

Names and definitions of terms:

- *TP* is *True Positives*, the number of relevant documents correctly returned.
- *FP* is *False Positives*, the number of irrelevant documents returned.

Again, notice that these ones are now whole numbers, not proportions.

The formula for calculating recall R is

$$R = \frac{TP}{TP + FN}$$

where FN is *False Negatives*, the number of relevant documents incorrectly rejected.

Because the question asks you to explain any new terms that appear in the formula for R, it's important to include both the term "False Negatives" and a description of what it means.

(c) The following tables give all the necessary figures for the calculation.

Hare	Relevant	Not relevant	Total
Retrieved	150	1050	1200
Not retrieved	50	2750	2800
Total	200	3800	4000

Tortoise	Relevant	Not relevant	Total
Retrieved	120	40	160
Not retrieved	80	3760	3840
Total	200	3800	4000

From these we can evaluate the performance of each system on this single query.

$$\textit{Hare}\text{ precision } P = \frac{150}{1200} = 1/8 = 0.125 \qquad \textit{Tortoise}\text{ precision } P = \frac{120}{160} = 3/4 = 0.75$$

$$\textit{Hare}\text{ recall } R = \frac{150}{200} = 3/4 = 0.75 \qquad \textit{Tortoise}\text{ recall } R = \frac{120}{200} = 3/5 = 0.6$$

This answers the question, which asks "Tabulate ... calculate ... Show your working". In particular, "tabulate" means to write out a table — so you need a table. Here, this *contingency table* with row and column totals makes it straightforward to calculate the answers: for each of them we need only pick out two values from the table and divide one by the other.

Some people used a different table showing "True" and "False" against "Positive" and "Negative". That turns out not to be so useful in calculating precision and recall. It's possible, but a bit more awkward as you'll find you need to add up a pair of diagonal elements rather than just picking out single values.

The formula for calculating the harmonic mean has many equivalent presentations. One of the simplest is

$$H = \frac{2PR}{P + R}$$

but you could also use

$$H = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} \quad \text{or} \quad H = \frac{2}{\frac{1}{P} + \frac{1}{R}} \quad \text{or} \quad \frac{1}{H} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right).$$

Using these we can calculate the harmonic mean of precision and recall for *Hare* as $3/14 = 0.21$ and for *Tortoise* as $2/3 = 0.67$.